Concept HDL User Guide

Product Version 14.2
January 2002

©1991-2002 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 1-800-862-4522.

All other trademarks are the property of their respective holders.

Restricted Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

Preface . . . 23
About This User Guide . . . 23
Finding Information in This User Guide . . . 23
Related Documentation . . . 24
Concept HDL . . . 24
Front-to-Back Flow . . . 25
Related Tools and Flows . . . 25
Typographic and Syntax Conventions . . . 25

2
About Concept HDL . . . 27
Concept HDL Features . . . 27
Concept HDL in the Design Flow . . . 29

3
What's New . . . 31
What's New in Concept HDL 14.2 . . . 31
What's New in Related Tools in 14.2 . . . 35
What's New in Concept HDL 14.0 . . . 37
What's New in Related Tools in 14.0 . . . 41

4
Getting Started . . . 43
Starting Concept HDL . . . 43
Concept HDL User Interface . . . 45
Menu Bar . . . 46
Toolbars . . . 47
Status Bar . . . 49
Console Command Window . . . 49
Context-Sensitive Menus . . . 49

Concept HDL Tasks . . . 50
Creating a Schematic . . . 50
Creating a Hierarchical Design . . . 52
Concept HDL Basics . . . 54
Where can I enter commands? . . . 54
Where are setup options? . . . 54
How do I pan drawings? . . . 55
How do I zoom in and out of a drawing? . . . 55
How do I customize Concept HDL? . . . 55
What commands can I use to edit schematic text? . . . 55
Are there menu shortcuts? . . . 56
How do I browse drawings and components? . . . 56
How do I add libraries? . . . 57
How do I add notes? . . . 57
How do I add parts? . . . 57
How do I connect parts? . . . 57
How do I name signals? . . . 57
How do I add properties? . . . 57
How do I add ports? . . . 57
How do I check my drawing for errors? . . . 58
How do I save a design? . . . 58
What is Page Locking? . . . 58
How do I generate a Verilog or VHDL netlist from a schematic? . . . 58
How do I add additional pages? . . . 58
How do I go to a specific page in a design? . . . 59
How do I plot a design? . . . 59
What are groups? . . . 59
What is different about working with groups? . . . 59
How do I locate parts and wires in a design? . . . 60
How do I generate a symbol view from a schematic? . . . 60
How do I package my design? . . . 60
How do I backannotate a design? . . . 60
How do I highlight objects in a design? . . . 60
How do I cross-reference a design? . . . 61
How do I archive a design? . . . 61
How do I display the online help? . . . 61

5
Project Creation and Setup 63

Introduction to Project Manager 63
Project Structure 64
Project Files 65
Local Project Files 66
Site Project File 67
Installation Project File 67
Project Flows 68
Creating a Project 68
Files Created for Your New Project 70
Creating a Site Project File 72
Setting Up a Project 74
Changing the Root Design for a Project 75
Creating a New Root Design for a Project 76
Editing the cds.lib File 76
Selecting Libraries for a Project 78
Adding Physical Part Table Files to a Project 79
Setting Up Tools 81
Specifying the Application Temp Directory 81
Selecting a Text Editor 82
Selecting a Property File 82
Setting Up a Log File 83
Selecting an Expansion Type 84
Selecting the Configuration for Expansion 84
Editing a Configuration 85
Creating a New Configuration View 86
Selecting Views for the Project 86

6
The Concept HDL Editing Environment 89

Setting Up Defaults 89
Setting Up Concept HDL Editor Options 89
Defining a Default Text Editor 90

Basic Editing Tasks . . . 91
Undoing an Operation . . . 91
Moving Objects . . . 92
Copying Objects . . . 94
Deleting Objects . . . 95
Changing the Color of Objects . . . 96
Drawing an Arc . . . 96
Drawing a Circle . . . 96
Splitting Overlaid Objects . . . 97
Displaying the Console Window . . . 97
Editing Text in Dialog Boxes and the Console Window . . . 98
Displaying Information . . . 98
Displaying Schematic Information . . . 99
Displaying Toolbars and Other Parts of the Concept HDL Window . . . 103
Highlighting Objects . . . 105
Turning Off Highlighting . . . 105
Opening the Markers Control Window . . . 105
Displaying the Markers Toolbar . . . 106
Displaying the Error Status Bar . . . 106
Basic Navigation in Concept HDL . . . 106
Panning the Drawing . . . 107
Zooming In and Out of the Drawing . . . 107
Navigating the Drawing Hierarchy . . . 108
Moving a Window . . . 109
Resizing a Window . . . 109
Closing a Window . . . 109
How do I navigate a design? . . . 109
Finding Nets and Cells in Your Design . . . 110
Navigating Nets in Your Design . . . 111
Exiting Concept HDL . . . 112
Running Commands with Strokes . . . 112
Guidelines for Strokes . . . 113
Modes in Concept HDL . . . 113
In Hierarchy Mode . . . 113
Expanded Mode . . . 114
Occurrence Edit Mode . . . 114

7
Creating a Schematic 117

Creating a Project 117
Starting Concept HDL 118
Creating a Design Page 119
Adding Page Borders 120
Adding Parts Using the Component Browser 122
Connecting Parts 125
- Drawing a Wire Manually 125
- Auto-Routing a Wire 125

Naming Signals 126
- Signal Naming Conventions 126
- Step Size in Signal Names 128

Adding Properties 130
Adding Ports 132
- Using PORT Symbols 132
- Rules for Using Port Symbols 133
- Port Association Restrictions 133

Working with Ports and Signals 134
- Setting the Initial Value of a Signal 135
- Specifying the Size of Nets 135
- Specifying the Assertion Level of Pins and Signals 136
- Creating an Alias for a Signal 137
- Declaring a Base Signal 138
- Global Signals 140
- Unnamed Signals 147
- Signal Concatenations 147
- Signal Replication 150
- Merge Symbols 150
- Signal Slices (Bit and Part Selects) 154
- Setting the Verilog Logic Type for Ports and Signals 157
- Setting the VHDL Logic Type for Ports and Signals 160
- Specifying Ranges for Ports, Signals and Aliases 163
- Unconstrained Ranges for Ports, Signals, and Aliases 165
- Resolved Types and Resolution Functions 165

Type Conversion . . . 166
Abstract Data Types in VHDL . . . 167
Using Iterated Instances . . . 167
X-Replication . . . 168
Using X-Replication . . . 169
Rules for Using X-Replication . . . 169
Saving a Design . . . 169
Working With Existing Designs . . . 172
Opening a Drawing . . . 172
Recovering a Drawing . . . 172
Reverting to the Previous Saved Version of a Drawing . . . 173

8
Working with Libraries and Components . . . 175

About the Standard Library . . . 175
Working with Libraries . . . 175
Adding New Libraries . . . 176
Browsing Libraries . . . 176
Adding Libraries to the Search Stack . . . 176
Removing Libraries from the Search Stack . . . 177
Defining Library Search Order . . . 177
Working with Components . . . 178
Browsing the Component List . . . 179
Creating Concept HDL Parts . . . 179
Creating a Symbol in Concept HDL . . . 180
Creating Entity Declarations from Symbols . . . 183
Creating the chips.prt File . . . 186
Creating a Part Table File . . . 187
Adding a Component . . . 188
Modifying Components . . . 189
Replacing a Component . . . 190
Defining Physical Property Options . . . 191
Breaking Up a Component . . . 194
Changing Pin States on a Component . . . 194
Choosing a Version of a Component . . . 195

Mirroring Components or Blocks . . . 195
Changing the Orientation of Components or Text . . . 195
Sectioning a Component . . . 196
Swapping Pins on a Component . . . 197
Ways to Determine if a Component Has Bus-Through Pins . . . 197
Deleting a Library Component (Cells, Views, and Files) . . . 197
Creating a SYNONYM . . . 197
Tapping a bit with a TAP symbol . . . 198
Creating a Page Border Symbol . . . 198
Customizing a Page Border in the Standard Library . . . 199
Creating a Page Border of Your Own . . . 200

9
Working with Wires . . . 205
About Signals and Connectivity . . . 205
About Bus Taps . . . 206
About Bus Names . . . 206
Drawing a Wire Manually . . . 206
Auto-Routing a Wire . . . 207
Stretching a Wire . . . 207
Bending a Wire . . . 208
Splitting a Wire . . . 208
Snapping a Wire to the Nearest Pin . . . 208
Naming a Signal . . . 208
Wiring Bus-Through Pins . . . 209
Marking Wire Connections . . . 209
Naming Signals on a Bus . . . 210
Specifying a Tap Symbol . . . 211
Attaching Values to Bus Taps . . . 212
Changing Wire Thickness and Pattern . . . 213

10
Working with Properties and Text . . . 215
About Properties . . . 215
Adding Properties . . . 216

Copying Properties . . . 216
Moving Properties . . . 216
Adding Properties . . . 217
Displaying and Modifying Property Attributes . . . 218
Making an Attribute File . . . 219
Adding Text . . . 220
Modifying Text . . . 221
Changing the Text Editor . . . 222
Adding Port Names from the Corresponding Symbol . . . 223
Swapping Notes or Properties . . . 223
Reattaching a Property from One Object to Another . . . 223
Specifying Property Display for Text . . . 223
Specifying Visibility of Pin Properties . . . 224
Specifying Visibility of Pin Numbers . . . 225
Case-Sensitivity in Property Names and Values . . . 226
Case Insensitive Property Values . . . 227
Case Insensitive Property Names . . . 228
Case Sensitivity and PPTs . . . 230
Constructing PPTs with Case-Sensitive Values . . . 231
Working with Text Macros . . . 232
About Text Macros . . . 232
How to use Text Macros . . . 232
Defining Text Macros on a Drawing Using the DEFINE Symbol . . . 233
Defining Text Macros Using \PARAMETER or \PARAM . . . 234
Defining a Text Macro in a File . . . 234
Working with Occurrence Properties . . . 236
About Occurrence Editing . . . 236
Viewing Occurrence Properties . . . 237
Occurrence Property File (OPF) . . . 237
Working with Custom Text . . . 239
Using Custom Text . . . 243

11
Working with Electrical Constraints . . . 247
What’s New in Constraint Management in Concept HDL 14.2 . . . 248

Using Designs from Previous Versions of Concept HDL . . . 251
Using Concept HDL 14.0 or 14.1 Designs in Concept HDL 14.2 . . . 251
Using Concept HDL 13.6 designs in Concept HDL 14.2 . . . 253
Traditional Flow . . . 260
Files Needed for Board Layout in Traditional Flow . . . 263
Constraint Manager Enabled Flow . . . 263
Files Needed for Board Layout in Constraint Manager Enabled Flow . . . 267
Moving from Traditional Flow to Constraint Manager Enabled Flow . . . 267
Starting Constraint Manager . . . 270
Working with Electrical Constraints . . . 272
Capturing Electrical Constraints . . . 272
Working with Electrical Constraints on Buses . . . 275
Working with Electrical Constraints on Pin-Pairs . . . 282
Working with Electrical Constraints in Hierarchical Designs . . . 287
Displaying Electrical Constraints Captured in Constraint Manager on the Schematic . . . 291
Modifying Electrical Constraints . . . 293
Deleting Electrical Constraints . . . 295
Auditing Obsolete Objects . . . 296
Viewing Constraint Differences between the Schematic and the Board . . . 297
The Constraints Differences-Logical Window . . . 299
The Constraints Differences-Physical Window . . . 300
Synchronizing Electrical Constraints between Schematic and Board . . . 302
Synchronizing Constraints in the Board with Constraints in the Schematic . . . 302
Synchronizing Constraints in the Schematic with Constraints in the Board . . . 307
Backannotating Constraints into the Schematic . . . 311
Cross Probing between Concept HDL and Constraint Manager . . . 312
Exiting Constraint Manager . . . 313
How Nets Are Displayed in Constraint Manager . . . 314
Files Created by Constraint Manager . . . 315
pstcmback.dat . . . 315
concept2cm.log . . . 315

12
Working with Block Designs 317
About Blocks 317
About View Generation in Hierarchical Designs 317
Generating Views for Top-Down Design 318
Generating Views for Bottom-Up Design 318
Creating Hierarchical Designs 318
Top Down Method 319
Bottom Up Method 321
Adding a Block 321
Using Read-only Blocks in Your Design 324
Navigating the Drawing Hierarchy 324

13
Working with Groups 331
Creating a Group by Rectangle 331
Creating a Group by Polygon 332
Creating a Group by Specifying an Expression 332
Creating a Group by Including Objects 333
Grouping the Entire Schematic 333
Including Additional Objects in a Group 333
Excluding Objects from a Group 333
Setting the Current Group 334
Viewing Group Contents 334
Moving a Group 334
Rotating a Group 335
Rotating a Group of Properties 335
Spinning a Group 335
Mirroring a Group 335
Copying a Group 338
Deleting a Group 339
Specifying Color for a Group 339
Highlighting a Group on the Schematic 339
Replacing Components in a Group 339

Replacing Component Symbols in a Group (Versioning) . . . 340
Breaking Up Components in a Group . . . 340
Specifying Property Display for a Group . . . 340
Changing the Text Size in a Group . . . 341
Modifying Components with the Same Part Name in a Group . . . 341

14
Working with Designs . . . 343

Expanding Your Design . . . 343
Finding Nets and Cells in Your Design . . . 344
Navigating Nets in Your Design . . . 346
Running Scripts . . . 347
Running a Script . . . 348
Stopping a Script . . . 349
Sample Scripts . . . 349
Highlighting (Cross-Probing) Objects . . . 350
Distributing Design Changes between Physical and Logical Designs . . . 350
Applying Connectivity Changes from the Physical Design to Your Schematic . . . 351
Back Annotating Your Design . . . 351
Module Ordering . . . 351
Changes in Module Ordering from Concept HDL 14.0 to Concept HDL 14.2 . . . 352
Using Module Ordering . . . 353
Excluded Modules File . . . 357
Displaying and Working with Schematic Page Numbers . . . 358
Displaying Page Numbers in a Schematic . . . 359
Modifying Custom Text for Page Numbers . . . 362
Updating Custom Text Variables for Page Numbers . . . 363
Renumbering Schematic Pages . . . 364

15
Netlisting Your Design . . . 371

Netlisting for Packaging the Design . . . 372
Disabling Netlisting of Designs . . . 374
What Happens if Netlisting is Disabled and You Package the Design? . . . 375
Netlisting for Simulation . . . 375

Netlisting for Digital Simulation . . . 376
Netlisting for Synthesizing a Design in Synplify . . . 378
Netlisting for Analog and Mixed Signal Simulation . . . 382
Netlisting Read-only Blocks Used in a Design . . . 384

16
Plotting Your Design . . . 387

Windows Plotting on Windows and UNIX Platforms . . . 387
Setting Up Windows Plotting Options . . . 388
Previewing the Design . . . 392
Plotting the Design . . . 394
Plotting in Batch Mode . . . 397
HPF Plotting on UNIX Platforms . . . 402
Setting up HPF Plotting Options . . . 403
Plotting the Design . . . 408
Plotting the Design from the Console Window . . . 410
Plotting Drawings from the UNIX Shell . . . 418
Hierarchical Plotting . . . 424
Hierarchical Plotting in Hierarchy, Expanded, and Occurrence Edit Modes . . . 424
Changing the Order in Which Designs Are Plotted . . . 426
Plotting Hierarchical Designs . . . 429
Customization of Plotting on UNIX Platforms . . . 433
Customization of Windows Plotting on UNIX Platforms . . . 433
Customization of HPF Plotting on UNIX Platforms . . . 436
Frequently Asked Questions in Plotting . . . 437

17
Cross-Referencing Your Design . . . 449

Overview . . . 449
About Cross References . . . 450
Types of Cross References . . . 450
How CRefer Cross-References a Design . . . 455
Cross-referencing a Design . . . 455
Viewing Cross References in Concept HDL . . . 457
Controlling CRefer Annotations Using UI Options . . . 458

Summarizing In-place Cross Referencing and Place Holder Support 459
Getting Started with CRefer 461
Preparing the Design for Cross Referencing 461
Determining the Right Cross Referencing Options 463
Creating the Cref Data File for Page Borders 464
Creating Custom Offpage I/O Flag Bodies 465
Making Cross References Permanently Visible 467
Adding Ports or Offpage Symbols to Signals 468
Using CRefer 468
Cross-Referencing the Design 469
Generating Cross References for a Design 470
Changing the Cref Data File 471
Configuring Run and Write Options 472
Configuring Formatting Options 474
Defining Output Reports 476
Deleting Cross References 478
Understanding CRefer Output 479
Identifying Inputs and Outputs 479
Sample Signals Labeled with Cross References 481
Signals that are Not Cross-Referenced 482
Reference Information 483
CRefer Text Reports 483
Schematic Reports 484
Cref Data File 484
I/O Types 486
Page Numbering 487
Support for Occurrence Property Data 487
Support for Concept HDL Custom Variables 487
Performing To/From Property Annotation 489
Design Sheet Variable Support 491
Placeholder Support 495

18
Archiving a Project 499
Archiver Overview 499

Creating a New Archive 500
Opening an Archive 502

19
Design Techniques 507

Introduction 507
Flat Designs 507
Creating a Flat Design 508
Concurrent Engineering of Flat Designs 509
Considerations of Flat Designs 509
Structured Designs 510
Creating a Structured Design 510
Benefits of Structured Designs 515
Considerations of Structured Designs 515
Hierarchical Designs 515
Creating a Hierarchical Design 516
Creating Symbols 518
Editing Symbols 518
Benefits of Hierarchical Designs 522
Comparing Design Techniques 522

A
Console Command Reference 525

Add 529
Arc 530
Assign 530
Attribute 531
Auto 532
Auto Commands 532
Backannotate 534
Badd 535
Bpadd 535
Bindview 536
Bpdelete 536
Bpmove 537

Bprename . . . 538
Brename . . . 538
Broute . . . 539
Browse . . . 540
Bstretch . . . 540
Bubble . . . 541
Busname . . . 541
Bustap . . . 542
Bwire . . . 543
Change . . . 543
Check . . . 544
Circle . . . 546
Copy . . . 547
Dehighlight . . . 548
Delete . . . 549
Diagram . . . 550
Directory . . . 551
Display . . . 552
Dot . . . 553
Echo . . . 554
Edit . . . 554
Error . . . 555
Exclude . . . 555
Exit . . . 556
Filenote . . . 557
Find . . . 557
Get . . . 558
Gotosheet . . . 559
Grid . . . 559
Group . . . 560
Hardcopy . . . 561
Highlight . . . 562
IGnore . . . 564
Include . . . 564
Library . . . 565
Loadstrokes . . . 566

Mirror . . . 567
Modify . . . 567
Move . . . 568
Next . . . 569
Note . . . 570
Paint . . . 570
Pause . . . 572
Pinnames . . . 572
Pinswap . . . 572
Plot . . . 573
PPTAdd . . . 574
PPTDelete . . . 574
PPTEcho . . . 575
Property . . . 575
Quit . . . 576
Reattach . . . 577
Recover . . . 577
Redo . . . 578
Remove . . . 578
Replace . . . 579
Return . . . 579
Rotate . . . 580
Route . . . 581
s2l . . . 581
Scale . . . 582
Script . . . 583
Searchstack . . . 584
Section . . . 584
Select . . . 585
Set . . . 587
    set sticky_on and set sticky_off . . . 592
Show . . . 592
Signame . . . 594
Smash . . . 595
Spin . . . 595
Split . . . 596

Strokefile . . . 596
Swap . . . 597
System . . . 598
Tap . . . 598
Textsize . . . 600
Undo . . . 601
Unhighlight . . . 601
Unix . . . 602
Updatesheetvars . . . 603
Use . . . 603
Vectorize . . . 604
Version . . . 604
Vpadd . . . 605
Vpdelete . . . 606
Window . . . 606
Wire . . . 607
Write . . . 608
Zoom . . . 609

B
Nongraphical Concept HDL (nconcepthdl) . . . 611
Running nconcepthdl . . . 611

C
Using the Standard Library Symbols . . . 613
VHDL_DECS and VERILOG_DECS Symbols . . . 613
Properties on VHDL_DECS and VERILOG_DECS . . . 613
Customizing the VHDL_DECS or VERILOG_DECS Symbol . . . 614
Using a Page Border as a Declarations Symbol . . . 615
SYNOP_DEC Symbol . . . 615
DECLARATIONS Symbol . . . 616
HDL_DECS Symbol . . . 616
TAP Symbols . . . 616
TAP . . . 617
CTAP . . . 618

BIT TAP . . . 618
LSBTAP Symbol . . . 619
MSBTAP Symbol . . . 620
CONCAT Symbols . . . 621
Rules for Using CONCAT Symbols . . . 622
SYNONYM . . . 622
Rules for Using SYNONYM Symbols . . . 624
PAGE Borders . . . 624
ORIGIN . . . 625
DRAWING . . . 625
REPLICATE . . . 626
SUPPLY_0 . . . 627
SUPPLY_1 . . . 628
PIN NAMES . . . 630
FLAG . . . 631
NOT . . . 631
DEFINE . . . 632
SIM_DIRECTIVES . . . 632
Other Symbols . . . 632
Customizing Standard library Symbols . . . 632

D
Error Checking Features in Concept HDL . . . 633
Cross-View Checking . . . 633
Entity Declaration Checking for Instantiated Components . . . 636
Netlisting Errors . . . 636

E
Concept HDL Files . . . 685
System Initialization File . . . 685
Cadence Library File . . . 685
ASCII Design Data Files . . . 685
Binary Design Data Files . . . 689
Symbol Files . . . 690
Connectivity Design Data Files . . . 693

Occurrence Property File (OPF) 696

F
The template.tsg File 699
Format of template.tsg File 699
defcell 699
defSymbol 700
Sample template.tsg File 712
Modifying the template.tsg File 714
Adding Symbol Properties 714
Adding Pin Properties 715
Modifying Symbol Attributes 716
Changing Pin Positions 716

G
Using Concept HDL with PSpice A/D 717
Notes for Using Concept HDL with PSpice A/D 718

H
Glossary 719

Index 731

Preface

About This User Guide

The Concept HDL User Guide explains how to use the Concept HDL schematic editor.

This user guide assumes that you are familiar with the development and design of electronic circuits at the system or board level.

Finding Information in This User Guide

This user guide covers the following topics:

See...	For Information About...
Chapter 2, “About Concept HDL”	Introduction to Concept HDL
Chapter 3, “What’s New”	New features and improvements in Concept HDL version 14.2 and its related tools
Chapter 4, “Getting Started”	Getting started with Concept HDL
Chapter 5, “Project Creation and Setup”	Creating and setting up design projects
Chapter 6, “The Concept HDL Editing Environment”	Setting up and using the Concept HDL editing environment
Chapter 7, “Creating a Schematic”	Concept HDL schematic tasks
Chapter 8, “Working with Libraries and Components”	Concept HDL libraries and components
Chapter 9, “Working with Wires”	Working with wires in drawings
Chapter 10, “Working with Properties and Text”	Concept HDL properties and text
Chapter 11, “Working with Electrical Constraints”	Working with electrical constraints in Concept HDL

See...	For Information About...
Chapter 12, “Working with Block Designs”	Concept HDL block designs
Chapter 13, “Working with Groups”	Concept HDL schematic groups
Chapter 14, “Working with Designs”	Concept HDL design tasks
Chapter 15, “Netlisting Your Design”	Netlisting Concept HDL designs
Chapter 16, “Plotting Your Design”	Plotting Concept HDL designs
Chapter 17, “Cross-Referencing Your Design”	Cross-referencing Concept HDL designs
Chapter 18, “Archiving a Project”	Archiving Concept HDL projects
Chapter 19, “Design Techniques”	Concept HDL design techniques
Appendix A, “Console Command Reference”	Concept HDL console commands
Appendix B, “Nongraphical Concept HDL (nconcepthdl)”	Nongraphical Concept HDL
Appendix C, “Using the Standard Library Symbols”	Standard library symbols
Appendix D, “Error Checking Features in Concept HDL”	Error checking features in Concept HDL and description of some netlisting errors
Appendix E, “Concept HDL Files”	Concept HDL files
Appendix G, “Using Concept HDL with PSpice A/D”	Using Concept HDL with the PSpice A/D simulator
Appendix H, “Glossary”	Concept HDL glossary

Related Documentation

You can also refer the following documentation to know more about related tools and methodologies:

Concept HDL

- For learning Concept HDL, see *Concept HDL Tutorial*.
- For information about the SKILL interface to Concept HDL, see *Concept SKILL Reference*.

Front-to-Back Flow

- For information on the front-to-back flow for PCB design, see *Front to Back Methodology Guide*.
- For information on the Design Synchronization solution, see *Design Synchronization and Packaging User Guide* and *Design Synchronization Tutorial*.
- For information about packaging your design, see the *Packager-XL Reference*.
- For information on Design Variance solution, see the *Design Variance User Guide* and *Design Variance Tutorial*.

Related Tools and Flows

- For information on maintaining and modifying the Concept HDL digital libraries, see the *PCB Librarian Expert User Guide*, *PCB Librarian User Guide*, and *Concept HDL Libraries Reference*.
- For information on the digital simulation interface provided by Concept HDL, see *Concept HDL Digital Simulation User Guide* and *Concept HDL Digital Simulation Tutorial*.
- For learning the Concept HDL Programmable IC flow, see *Programmable IC Tutorial*.
- For information on capturing electrical constraints in Constraint Manager, see the *Constraint Manager Design Guide*.
- For information on Concept HDL data management, see *Design Manager User Guide*.
- For information on CheckPlus-HDL, see *CheckPlus User Guide*.
- For information about moving from SCALD to HDL flow, see *SCALD to HDL Evolution Guide*.
- For information on creating custom interfaces to translate the HDL database into a format that can be used by an external system and to update the HDL database with changes from a physical design system, see the *CAE Views Programming Guide*.

Typographic and Syntax Conventions

This list describes the syntax conventions used for this user guide:

`literal`	Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names.
`argument`	Words in italics indicate user-defined arguments for which you must substitute a name or a value.
`|`	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character.
`[ ]`	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
`{ }`	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.

2

About Concept HDL

Concept HDL is a design environment that supports behavioral and structural design descriptions captured in text and graphics. It incorporates block editing functions for quick architectural design.

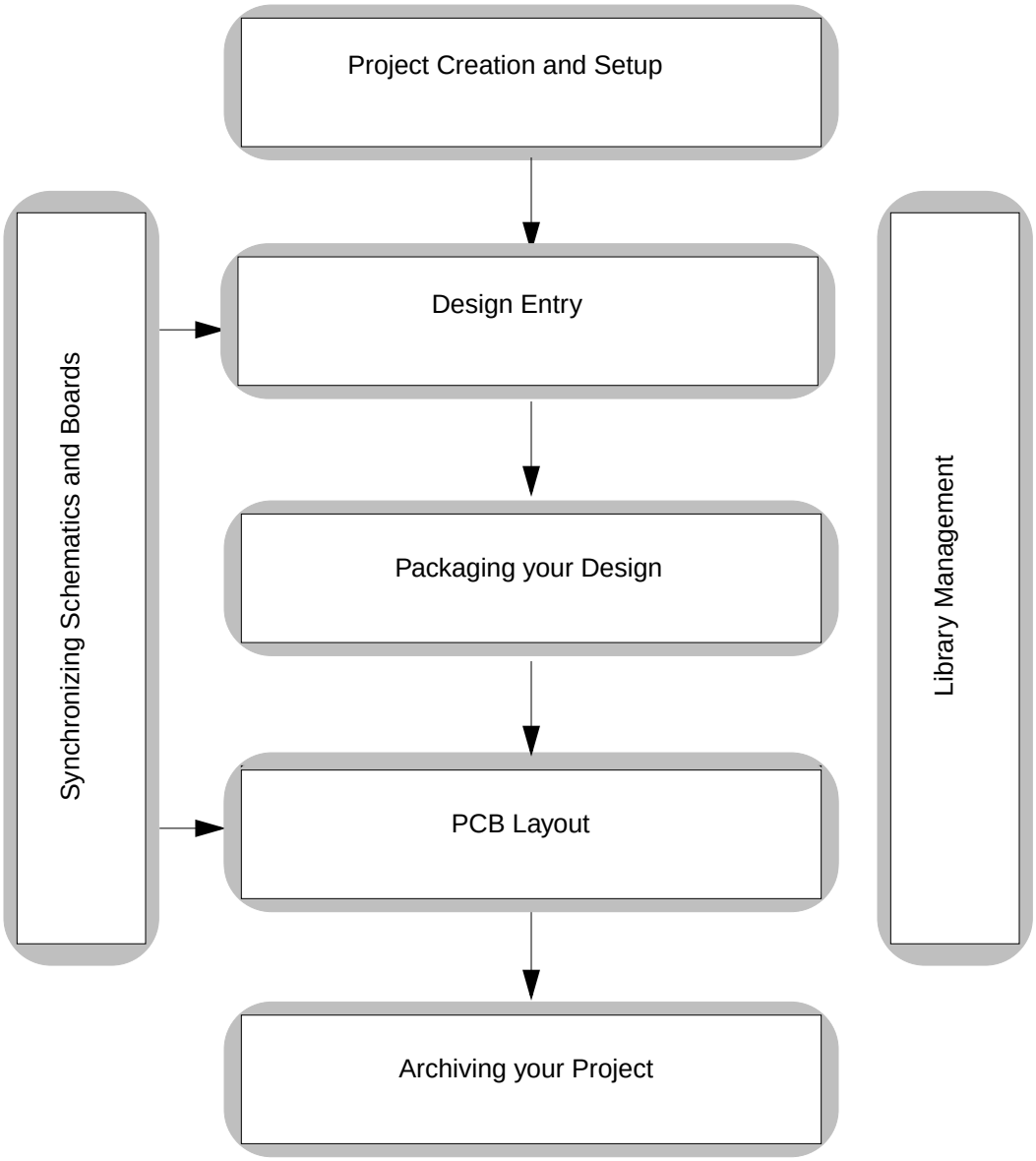
Concept HDL organizes schematic information into pages. It captures and displays only one page of schematic information at a time. Concept HDL is a by-reference editor because it references all parts in the schematic from various libraries that reside at the reference or local area. A standalone, self-contained database of the libraries and the design can be created using the Archiver utility.

Concept HDL Features

- A top-down (hierarchical) design that lets you quickly draw blocks and connect wires between blocks. A cross-view generator (`Genview`) to create blocks from HDL descriptions or automatically generate HDL text from high-level diagrams.
- A customizable user interface that lets you customize menus and toolbars, map keys to functions, and create new commands.
- A hierarchy editor lets you view the structure of your design.
- An attribute editor that lets you annotate properties on a design to drive the physical layout.
- Integration with the Design Synchronization toolset. This toolset lets you view differences between your schematic and the board layout and then synchronize them.
- Cross-probing between Concept HDL and other Cadence tools.
- Support for design reuse. You can associate logical components with a layout section to create reusable components. This component can be reused in other areas in your design and also in the designs you create later.
- Integration with CheckPlus, an advanced rule checking and rule development system.

- Integration with Constraint Manager tool that allows you to capture and manage electrical constraints as you implement logic.
- Support for importing Intermediate File Format (IFF) files that can be created for radio-frequency (RF) designs. You can create radio-frequency (RF) designs using tools such as ADS or MDS by Agilent Technologies, Inc. The ADS tool supports the creation of Intermediate File Format (IFF) files. Once the RF design is ready, you can create IFF files for the schematic and layout of the design. You can import a schematic IFF file into Concept HDL to transfer the graphics and connectivity data of the RF design into Concept HDL and then use the RF design as a block in a larger Concept HDL design. For more information, see Importing *Radio-Frequency Designs* in the Concept HDL online help.
- Concept SKILL, the SKILL programming interface to Concept HDL.

Concept HDL in the Design Flow



3

What's New

The new features in Concept HDL and related tools are described below:

What's New in Concept HDL 14.2

Improved Integration with the Constraint Manager Tool

In this version, the integration between Concept HDL and the Constraint Manager tool has been improved to provide the following features:

- Support for synchronization of electrical constraints in Constraint Manager and Concept HDL
- Improved support in Design Synchronization flow for importing and exporting constraints if you are using Constraint Manager with Concept HDL
- Support in Design Differences tool to display report of differences in electrical constraints information between schematic and board.
- Support for quickly adding placeholders for electrical constraints in the schematic
- Syntax checking of electrical constraint properties in the schematic when you start Constraint Manager from Concept HDL or run Export Physical

For more information, see the "Working with Electrical Constraints" chapter of the *Concept HDL User Guide*.

Enhancements in HPF Plotting Mode

The following enhancements have been made in the HPF Plotting mode in Concept HDL:

- Hierarchical Plotting in HPF Plotting Mode

  In Concept HDL 14.0, hierarchical plotting was supported only in the Windows plotting mode. Concept HDL 14.2 supports hierarchical plotting in the HPF Plotting mode also.

You have the option to plot all the schematics in a design hierarchy or to exclude some or all occurrences of blocks from the hierarchy of a design.

- Plotting of Occurrence Properties and Custom Text

  In this version, the HPF Plotting mode supports plotting of occurrence properties in the Occurrence Edit mode and plotting of custom text in Hierarchy, Expanded and Occurrence Edit modes.

- Usability Enhancements in HPF Plotting Mode

  In this version, the following usability enhancements have been made in the HPF Plotting mode:

  - In Concept HDL 14.0, users had to enter the plotter name in the *Plotting* (HPF) tab of the *Concept Options* dialog box. In Concept HDL 14.2, the *Plotting* (HPF) tab displays the list of plotters by reading the `.cdsplotinit` file.
  - In Concept HDL 14.0, users had to enter the font name in the *Plotting* (HPF) tab of the *Concept Options* dialog box. In Concept HDL 14.2, the *Plotting* (HPF) tab allows users to select the supported fonts from a drop-down list.

- Performance Enhancements in HPF Plotting Mode

  The performance of plotting in the HPF Plotting mode has been improved in Concept HDL 14.2.

Support for Displaying and Working with Page Numbers in Hierarchical Designs

In Concept HDL 14.0, you could use the <CON_PAGE_NUM> and <CON_TOTAL_PAGES> custom text variables to display the page numbering information for pages in a cell.

In Concept HDL 14.2, you can use the <CURRENT_DESIGN_SHEET> and <TOTAL_DESIGN_SHEETS> custom text variables to display the page numbering information on pages in the entire design hierarchy. For example, if the root design A in a hierarchical design has three pages, and sub design B has four pages, the value of:

- CURRENT_DESIGN_SHEET variable will be 1 for the first page of the root design A and 3 for the last page in the root design A. The value of the CURRENT_DESIGN_SHEET variable for the first page of sub design B will be 4 and the value of the variable for the last page of sub design B will be 7.
- TOTAL_DESIGN_SHEETS variable will be 7. The TOTAL_DESIGN_SHEETS variable takes into account the actual number of pages in a hierarchical design. Even if the highest page number assigned to a page in a hierarchical design is greater than the

actual number of pages in the design, the `TOTAL_DESIGN_SHEETS` variable will display only the actual number of pages in the design.

You can choose *File > Edit Page/Symbol > GoTo*, select the *Calculate page number in hierarchy* check box and enter the page number to open a schematic page in a hierarchical design. You can also use the `gotosheet <page_number>` console window command to open a schematic page in a hierarchical design.

For more information, see the "*Working with Designs"* chapter in the *Concept HDL User Guide*.

Support for Allowing Global Signals to be Shorted in your Design

If a global signal in your design is shorted with another global signal, error messages are displayed when you save or package the design.

You might have intentionally shorted some global signals in your design. If you want to allow such global signals to be shorted, add them in the *Allowed Global Shorts* list in the *Output* tab of the *Concept HDL Options* dialog box. No error messages for global signal short are displayed if the pairs of global signals listed in the *Allowed Global Shorts* list are shorted.

For more information, see "*Shorting of Global Signals*" in the Creating your Schematic chapter of the *Concept HDL User Guide*.

Support for Declaring Base Signal Names

When two signals are aliased or synonymed, Concept HDL selects one of the signal names as the base signal. The name of the base signal becomes the name of corresponding physical net in Allegro.

You may want the name of a particular aliased or synonymed signal to be passed to Allegro as the physical net name. Concept HDL allows you to do this by declaring a particular aliased or synonymed signal as a base signal.

For more information, see "*Declaring a Base Signal*" in the Creating your Schematic chapter of the *Concept HDL User Guide*.

Enhancements in Module Ordering

In Concept HDL 14.0, if there are multiple occurrences of a module in a design, the *Module Ordering* dialog box displays only one occurrence of the module in Hierarchy or Expanded mode and all occurrences of the module in Occurrence Edit mode. In Concept HDL 14.2, the

Module Ordering dialog box displays all occurrences of the module in Hierarchy, Expanded and Occurrence Edit mode.

Page Locking

Concept HDL locks a schematic page that is being edited by a user. If another user opens the same page for editing, Concept HDL displays a message that the page is locked by the first user and that any changes he makes in the page cannot be saved.

This feature prevents users from unknowingly overwriting changes made by each other.

Usability Enhancements

The following usability enhancements are available in Concept HDL 14.2:

- Select an object on the schematic and drag the mouse to move the object.
- Press the left mouse button and drag the mouse to select multiple objects on the schematic. You can now click on one of the selected objects and drag the mouse to move all the selected objects.

Note: To use the above-mentioned usability enhancements, deselect the *Ctrl+LMB Select and Drag* check box in the *General* tab of the *Concept Options* dialog box.

- Support for rotating a group of properties

  You can rotate a group that consists only of properties. The properties are rotated in place and not as a whole group.

Split Vector Ports Option in Genview

Genview now allows you to split vectored ports in the source view or source file into multiple pins (representing each bit of the vectored port) on the symbol.

For example, if the source view or source file has a vectored port `DATA<3..0>`, the following four pins will be added on the symbol:

- DATA<3>
- DATA<2>
- DATA<1>
- DATA<0>

What's New in Related Tools in 14.2

Enhancements in Design Synchronization Flow

- Backannotation from Import Physical and Export Physical

  You can backannotate the schematic from both Import Physical and Export Physical. This saves you the need to go to Concept HDL and run *Tools > Backannotate* to backannotate the changes made in the layout into the schematic. You can even backannotate the changes in the Occurrence Edit mode.

- User Interface (UI) Changes

  In the Packager Setup - Properties tabbed sheet, a new list box named *Property Conflicts Filter* is added. If you add properties in this list, the properties are filtered from the `pstprop.dat` file. In the Packager Setup - Subdesign tabbed sheet, a new field *Subdesign Suffix Separator* is available. You can use this field to define a different character for renaming reference designators for reuse modules. By default, the underscore letter (_) is used to define reference designators for reuse modules.

Changes in Packager-XL

Packager-XL has been enhanced to provide the following new features:

- OPF Performance Optimization

  In Version 14.0, Packager-XL used to dump many redundant properties into the OPF as they were also present in the `pxl.state` file. In Version 14.2, Packager-XL does not write these properties into the OPF. As a result, there has been a significant increase in Packager-XL performance.

- New Packager-XL Directives

  Three new directives `REGENERATE_PHYSICAL_NET_NAME`, `FILTER_CONFLICTING_PROP`, and `SD_SUFFIX_SEPERATOR` are added to Packager-XL.

  REGENERATE_PHYSICAL_NET_NAME—You can use this directive to delete all existing physical net names and generate them afresh. The default value of this directive is `off`. The use of the `REGENERATE_PHYSICAL_NET_NAME` directive will ensures that the changes done on net names in Allegro are not lost during successive packaging.

  FILTER_CONFLICTING_PROP—If you add properties in this directive, those properties are filtered from the `pstprop.dat` file. From the UI, you can access this directive by adding properties in the *Property Conflicts Filter* list of the Packager Setup

- Properties tabbed sheet. The use of the FILTER_CONFLICTING_PROP directive ensures that you can place only relevant properties in the pstprop.dat file.

SD_SUFFIX_SEPERATOR—You can use this directive to define a different character for renaming reference designators for reuse modules. By default, the underscore letter (_) is used to define reference designators for reuse modules. From the UI, you can access this directive by defining a character in the *Subdesign Suffix Separator* field in the Packager Setup - Subdesign tabbed sheet.

- Improved Refdes Locking

  The REUSE_REFDES directive was introduced in Version 14.0 of Packager-XL. This directive is used to control the reuse of reference designators for changed or deleted components. However, in Version 14.0 to use this directive you were required to run Import Physical and generate feedback files from Allegro. In Version 14.2, Packager-XL allows you to reuse reference designators without the need to generate feedback files. Therefore, you are not required to run Import Physical to ensure that the reference designators for changed or deleted components in Allegro are reused.

- Design Reuse Tutorial

  A new tutorial named *Design Reuse Tutorial* is available on CDSDoc. This tutorial explains the process of creating reused blocks and using them in different designs. The tutorial also includes a detailed Frequently Asked Questions (FAQs) section that explains different issues around successful design reuse in a user's environment.

- Improved Error Message Documentation

  The *Error Message* section in the Cadence document *Packager-XL Reference* in CDSDoc is improved. Descriptions of approximately errors and warnings are available. These messages include examples where errors occurred and possible workarounds to deal with these errors.

Improvements in BOM-HDL

- Better Report Formatting

  BOM-HDL now has the following usability features:

  - In Version 14.0, the BOM_PART property was a mandatory column in the BOM report. In Version 14.2, you can remove BOM_PART from a BOM report. This column will be mandatory only if you do not check any other column.
  - The BOM report lists a column header that lists the names of properties displayed. You can remove this header row, by clearing the Print Column Header check box in the BOM-HDL Physical Part Specifications tabbed sheet. This will help you to easily import BOM reports in Microsoft Excel.

- BOM-HDL sets the width of the variant column according to the Part Number. This prevents the wrapping of any text in multiple lines for same PPT row.
- The Add Header parameter dialog box allows you to specify five new header properties—Product, Version, Description, Project Path, and Variant.

- Command Line BOM Generation Improvements

  The command-line BOM generation has no UI call. When the report is generated, a "Report successfully generated" message is displayed on the console. BOM-HDL provides complete interoperability of paths for the output file. You can use absolute and relative paths, and generate outputs across platforms. For example, you can run the tool in Windows and generate the output on Solaris.

Enhancements in Concept HDL Digital Simulation Flow

- Support for VHDL Map Files

  In the 14.2 release, Concept HDL simulation flow provides a complete map file solution. From this release, both the Verilog map file and the VHDL map files are supported.

- Asymmetrical Part Support

  You can now use the Concept HDL digital simulation interface to simulate split parts and asymmetrical parts. When in a part having a large pin count is represented using multiple symbols, then each symbols is called the split part for the part. You can now simulate such parts using properties such as, `SPLIT_INST` and `SPLIT_INST_NAME`.

- Instance-Specific Binding using Verilog Simulators

  Two new properties, `MODEL_DIR` and `MODEL_FILE`, have been added to support instance-specific binding in the Verilog-XL flow. The `VERILOG_LIB` property is used for instance-specific binding in the NC Verilog flow.

What's New in Concept HDL 14.0

New Menu Use Model

Concept HDL has historically supported a post select model for performing schematic operations. This means that the command from the menu, command line or toolbar, is selected first and then the objects in the schematic are selected.

For example, to rotate a component using the post select model, first choose the menu option *Edit > Rotate* and then click on the component.

1. Choose *Edit > Rotate*.
2. Click on the component.

In the 14.0 release, Concept HDL supports a pre select model. Using this feature, you can select the object first and then the menu option for the operation you want to perform.

To rotate a component using the pre select model, first click on the component and then choose the menu option *Edit > Rotate*.

1. Select the component.
2. Choose *Edit > Rotate*.

The post select menu use model is the default model. You can set the preferred menu use model by selecting the *Tools > Options > General > Enable Pre Select Mode*.

Multiple Object Selection and Deselection

You can select multiple objects on a page in Concept HDL by keeping the *SHIFT* key pressed and clicking on the objects you wish to select.

A selected object can be deselected by keeping the *SHIFT* or *Ctrl* key pressed and clicking on the objects you wish to deselect.

Context-Sensitive Menus

Every object in Concept HDL has a context-sensitive menu attached to it. The menu appears when you right-click on the object. The menu contains menu options to perform certain operations that are relevant to the current object and context. Examples of operations on a symbol are copy, delete, edit, and rotate, to mention just a few.

Hierarchical Plotting

Plotting in Concept HDL was always done for a design, a page, or a set of pages. Wildcards could be used to plot multiple designs, but never in the context of a single project. This limitation is now removed with the addition of a hierarchical plotting feature. You have the option to plot all the schematics in a design hierarchy, or to exclude some or all occurrences of schematics from the hierarchy of a design.This new feature is provided from the *Plot* dialog box.

Although the Hierarchical plotting feature is supported in the Hierarchy, Expanded and Occurrence Edit modes, occurrence properties are plotted only in Occurrence Edit mode.

Module Ordering

With the addition of the Hierarchical Plotting solution comes the requirement to control the ordering in which designs are plotted. You can change the order in which a hierarchical design is plotted and cross-referenced using a new graphical interface available from the *Edit > Module Order* menu option. This also supports user selection of modules to exclude from cross referencing or plotting.

Automatic Page Border Placement

Concept HDL automatically places a page border every time you open a new page for creating a schematic. By default, this feature is off. To turn it on, select a page border in the *General Options* section of Concept HDL Setup.

Custom Text and Variables

Schematic designs always have some level of documentation provided on each sheet. This is typically included in the page borders and includes data such as project name, project number, page number, total pages, engineer or designer name as well as a whole host of other company-specific data. Placing this information in the page border has always required either a custom page border for the design or a manual update of the data on each page. This is time-consuming and error-prone.

Now Concept HDL provides the ability to attach custom text to objects on a schematic. This custom text includes variables that are automatically evaluated by Concept HDL. Some variables are system-defined, such as page number and design name. Others are user-defined and located in the project file. Unlike notes, custom text must be attached to objects on the schematic.

Page Renumbering

Concept HDL now provides you the facility of renumbering the pages of a design. This facility has been provided through a number of console commands. You can swap two pages, move a page to a new location, or delete a page from a design.

Nongraphical Concept HDL

The executable `nconcepthdl` allows the core schematic editor to be run in a non-graphical mode. This allows you to run Concept HDL without a graphics terminal in the background. `nconcepthdl` is extremely useful for running batch processes, such as hardcopy or back annotate, without having to invoke the overhead of the graphical editor.

Support for Technology-Independent Libraries

Cadence now provides you with technology-independent libraries. These libraries radically reduce the disk space required for storing a library and make the entire solution more user-friendly.

Integration with the PSpice A/D Simulator

You can now simulate designs created in Concept HDL using the PSpice A/D Simulator.

PSpice A/D is a simulation program that models the behavior of a circuit. PSpice A/D simulates analog-only, mixed analog/digital, and digital-only circuits. Used with Concept HDL, PSpice A/D is much like a software-based breadboard of your circuit. You can use it to test and refine your design before manufacturing the physical circuit board.

For more information, see the *PSpice A/D User Guide*.

Component Wiring

There are two new features in Concept HDL for automating the component wiring process. The first feature allows a 2-pin component to be inserted into an existing net. Select the component in the *Add Component* form and place it on top of a wire. The pins will short, Concept HDL will recognize the short and remove the wire between the two pins thus splitting the net into two pieces. It is up to you to decide whether or not to name one of the two nets that are created. If an existing signal name is available, it is attached to the wire closest to its original attach point.

The second new feature is for quickly wiring multiple signals between components. Add a component like a memory device that contains a set of pins on each side. Then add another version of this component and place it in a manner that the pins are touching. When you move one of the components, you will notice that wires are formed between the pins.

New Concept HDL Tutorial

The new Concept HDL tutorial in this release guides you through the tasks of creating a simple flat schematic and a hierarchical schematic. It also demonstrates the use of many advanced features in Concept HDL.

What's New in Related Tools in 14.0

Property Flow Setup

You can use the *Property Flow Setup* dialog box to define the flow of properties between Concept HDL and Allegro.

You can define whether a property:

- Belongs only to the Concept HDL schematic and should not be transferred to the Allegro board.
- Belongs only to the Allegro board and is not backannotated to the Concept HDL schematic.
- Can be transferred between Concept HDL and Allegro.

You can access the *Property Flow Setup* dialog box in two ways:

- Choose the *Property Flow Setup* button in the *Packager Setup* dialog box.

  OR

- Choose *Difference > Property Flow Setup* in the *Design Difference* menu bar.

Changes in Packager-XL

Packager-XL has been enhanced to provide the following new features:

- Reference Designator Locking

  A new directive, named `REUSE_REFDES`, allows you to control the reuse of reference designators for changed or deleted components. By default, the existing reference designators for changed or deleted components are reused for new components. However, by setting the `REUSE_REFDES` directive to `off`, you can lock the existing reference designators and thereby cause new reference designators to be created for new components.

- Provide an output file of the property conflicts and logical net name changes that occur during packaging. These can be found in the `pstprop.dat` file after packaging.

- Text macro support on nets and pins

  Text macro support is now available for nets and pins. Packager-XL substitutes text macros for net and pin properties in the same way as it does for instances.

Improvements in the Cross Referencing Tool

The CRefer tool is improved over release 13.6 and it supports the following useful features:

- Performance Improvement

  CRefer can now cross reference large designs with substantial performance improvement.

- To or From Property Annotation

  CRefer performs to or from property annotation to specify from where the pages in an expanded design come from the original design, and where the pages from the original hierarchical design are included in the cross-referenced expanded design.

- Support for Occurrence Property Data

  CRefer can now annotate OPF data in the expanded mode.

- Reports

  CRefer can be used to produce new reports, such as a synonym report. This report traces a net across a hierarchical design. Besides new reports, the existing CRefer reports are also improved. For example, the basenet and nets-by-page reports include the direction symbols with the signal and synonym information. The CRefer part report is enhanced, and it includes the design name and instance identifier. You can also use the CRefer tool to append all reports at the end of the root schematic.

- Support for Module Page Reordering in Concept HDL

  You can use the Module Ordering function in Concept HDL to control the ordering of the modules in the flattened view for a hierarchical design.

- Top level CRef Annotation

  Cross-references are also annotated at the top level in a hierarchical design.

For more information about the improvements in the CRefer tool and how to use the tool, refer *CRefer Online Help*.

4

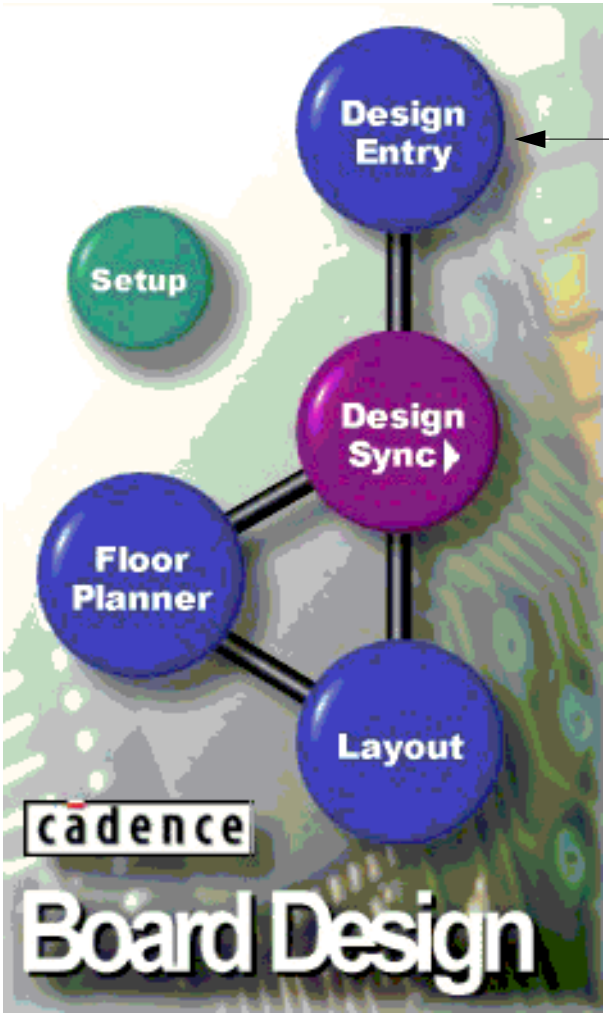
Getting Started

This chapter contains the following information:

- Starting Concept HDL on page 43
- Concept HDL User Interface on page 45
- Concept HDL Tasks on page 50
- Concept HDL Basics on page 54

Starting Concept HDL

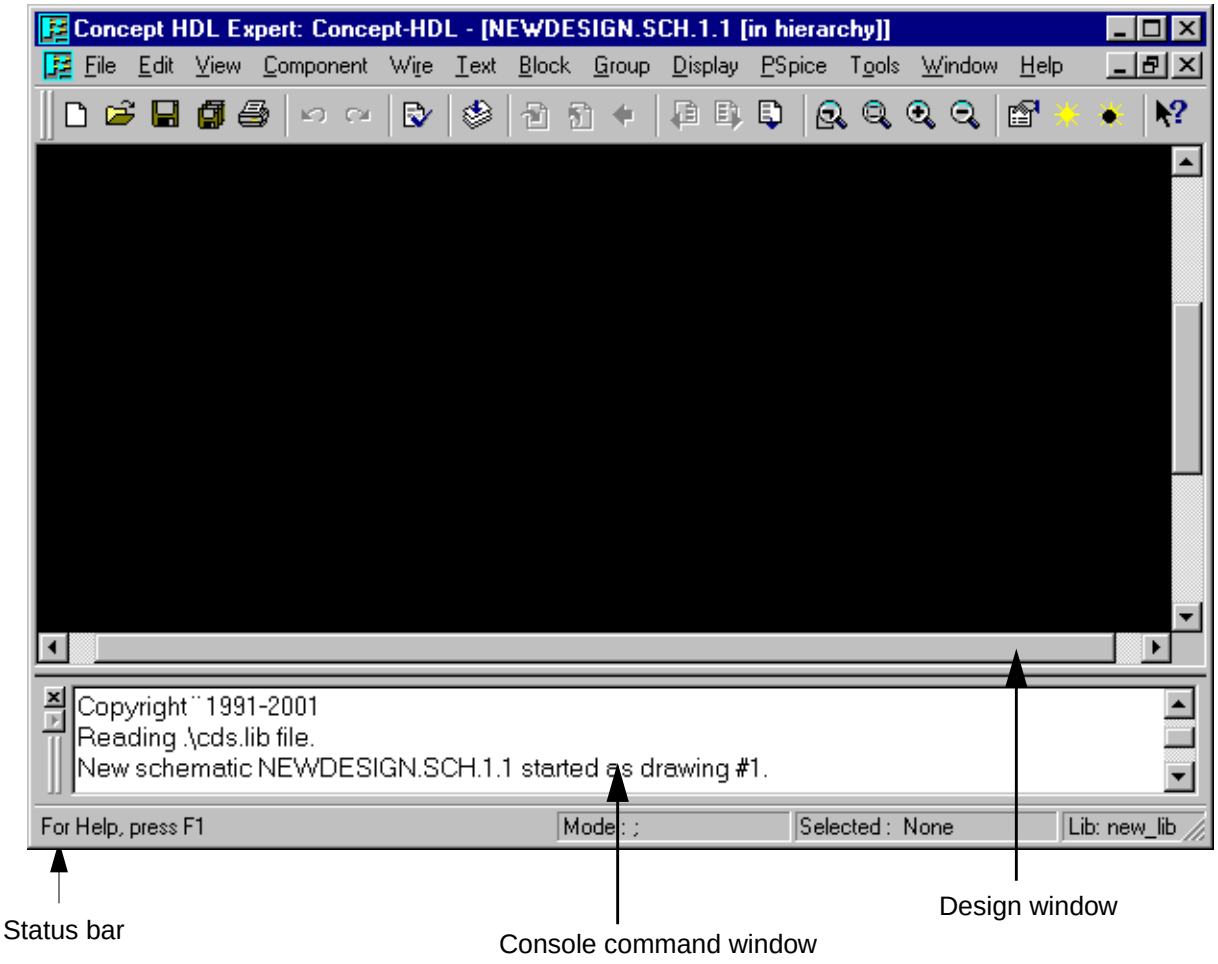
After you open the desired design project in Project Manager, the flow area of Project Manager displays the Cadence Board Design flow. In the Board Design flow, click the *Design Entry* icon.



Click to invoke Concept HDL

Concept HDL User Interface

When you click the *Design Entry* image in Project Manager, the Concept HDL user interface appears (as shown in the following figure).


Concept HDL Expert: Concept-HDL - [NEWDESIGN.SCH.1.1 [in hierarchy]]
File Edit View Component Wire Text Block Group Display PSpice Tools Window Help
Copyright ¨ 1991-2001
Reading .\cds.lib file.
New schematic NEWDESIGN.SCH.1.1 started as drawing #1.
For Help, press F1
Mode: ;
Selected : None
Lib: new_lib
Status bar
Console command window
Design window

The interface consists of the following elements:

- Design window
- Menu bar
- Toolbars
- Status bar
- Console command window

- Context-sensitive menus

Menu Bar

The Concept HDL menu bar includes the following menus:

- File

  For operations such as opening, saving, and plotting a drawing.

- Edit

  For operations such as Undo, Copy, Paste, Delete, Spin, and Color.

- View

  For operations such as Zoom, Pan, and Grid.

- Component

  For operations that can be done on a part such as adding, replacing, and modifying a part.

- Wire

  For operations such as connecting parts and naming signals.

- Text

  For operations such as adding properties and notes.

- Block

  For operations such as adding blocks.

- Group

  For operations such as creating groups and performing editing functions on groups.

- Display

  For operations such as highlighting and dehighlighting components.

- PSpice

  For performing analog, digital and mixed-signal simulation using the Pspice A/D simulator. This menu is visible only if you have installed PSpice A/D.

- Tools

  For operations such as setting up defaults, customizing, updating the schematic with layout changes, updating the layout with schematic changes, finding nets and instances in your design, global navigation, checking your design, and running scripts.

- Window

  For operations such as opening a new window, cascading and tiling it.

- Help

  Invokes Online Help, Known Problems and Solutions, and Product Notes.

Toolbars

Concept HDL has seven toolbars.They are Standard, Add, Block, Edit, Group, Markers, and Color. If you have installed PSpice A/D, the following six additional toolbars are available. For more information on these toolbars, see the PSpice User's Guide.

- Analog
- Passive
- Source
- Linear
- Discrete
- Misc

Standard Toolbar

The *Standard* toolbar has the standard functions that operate on a drawing (Open, Save, Print, Zoom, Previous Page, Next Page, Check and so on).

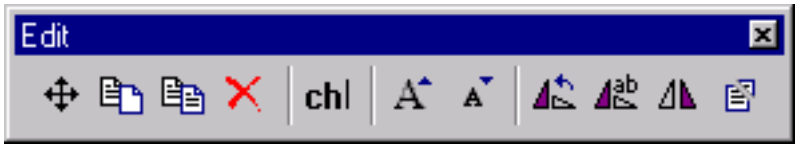
Block Toolbar

The Block toolbar lets you add blocks, add pins on blocks and draw wires to connect blocks.

Add Toolbar

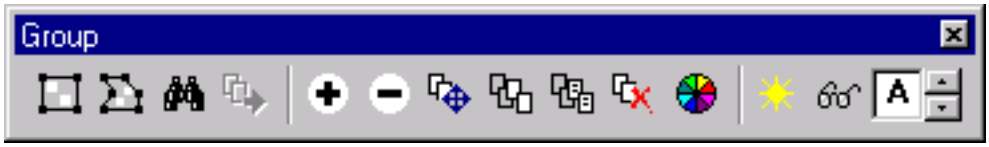
The *Add* toolbar lets you add objects (components, wires, and text) and graphics such as dots and circles.

Edit Toolbar



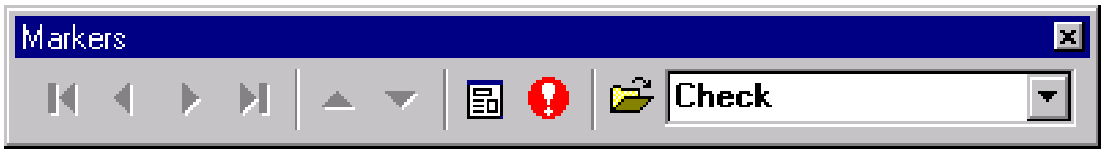
The *Edit* toolbar lets you perform edit operations such as copy, paste, delete, and spin.

Group Toolbar



The *Group* toolbar has all the commands for creating and modifying a group. A group is a collections of objects such as notes, components, wires, and properties.

Markers Toolbar



The *Markers* toolbar helps you traverse through schematic errors.

Color Toolbar

The *Color* palette lists the colors supported in Concept HDL and allows you to quickly change the colors of various objects.

Status Bar

The status bar displays a single line about the action you are performing or when Concept HDL expects you to perform an action.

Console Command Window

You can type commands in this window. The window can also be used to manually test any scripts that you have written for Concept HDL. To enable or disable the Console Command window, choose *View > Console Window*.

Context-Sensitive Menus

Every object in Concept HDL has a context-sensitive menu attached to it. The menu appears when you right-click on the object. The menu contains options to perform certain operations that are relevant to the current object and its context. Examples of operations on a symbol are copy, delete, edit, and rotate.

Concept HDL Tasks

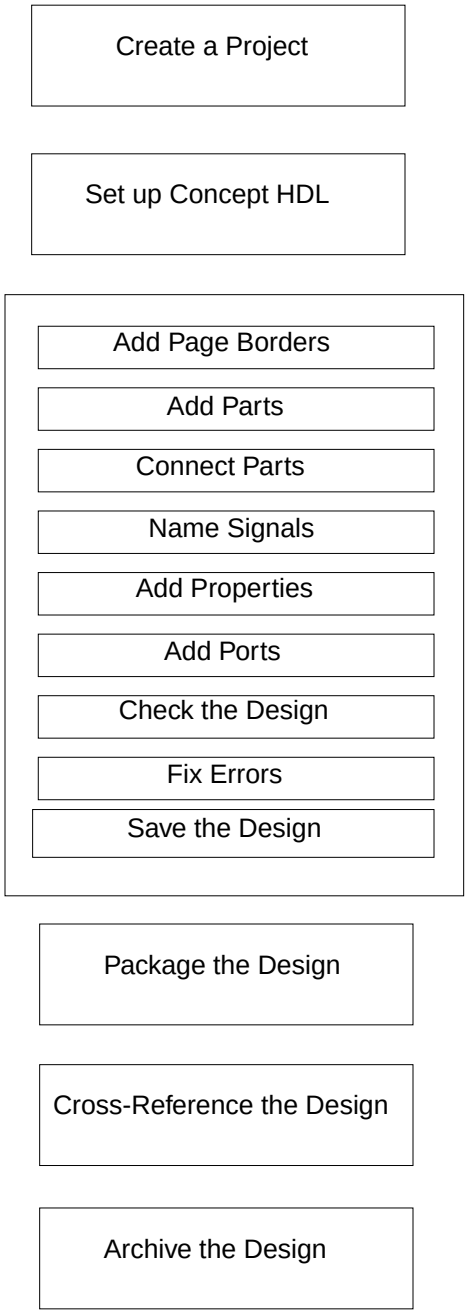
The Concept HDL tasks covered in this section are:

- Creating a Schematic on page 50
- Creating a Hierarchical Design on page 52

Creating a Schematic

The following figure illustrates the sequence of tasks you perform in Concept HDL to create a schematic.

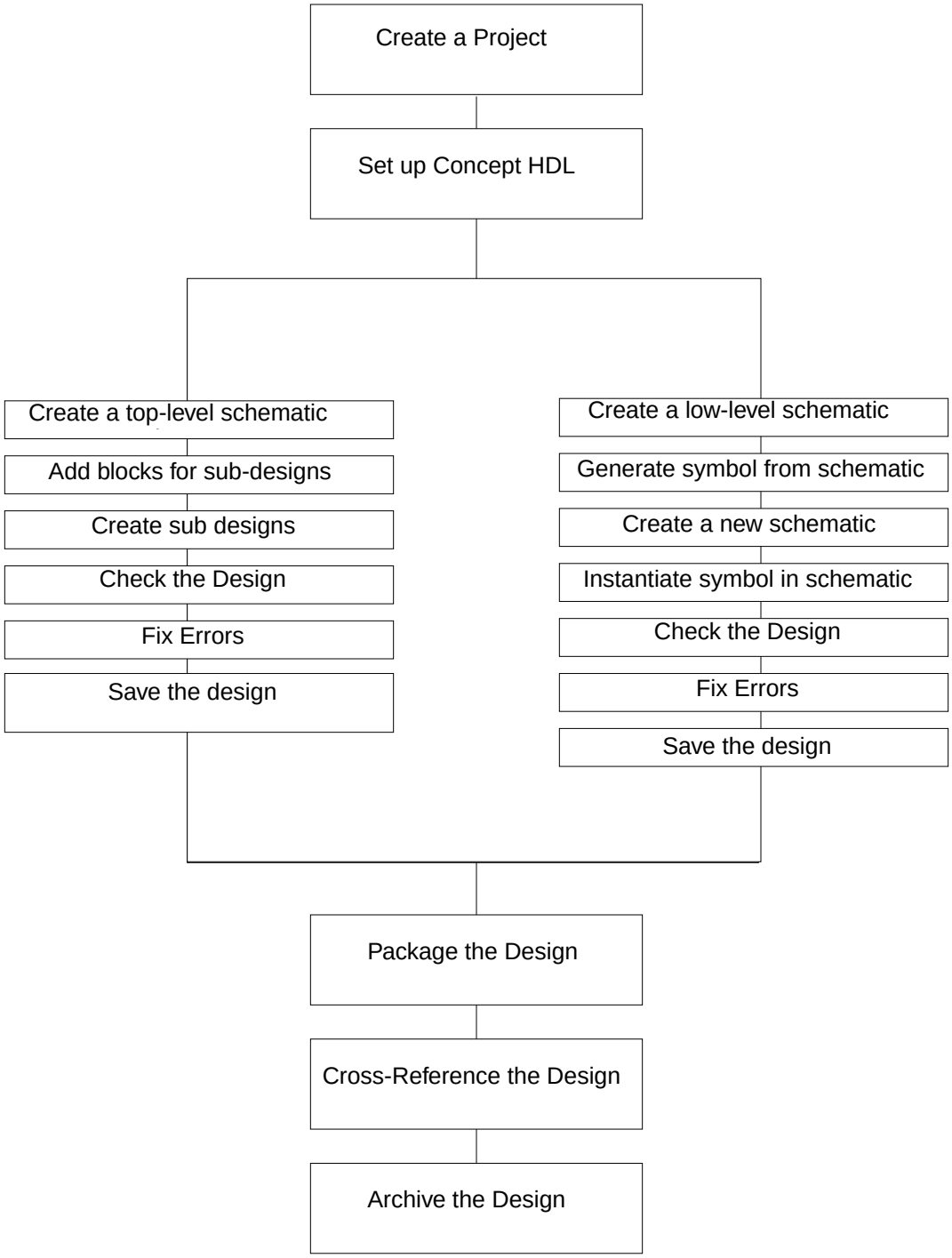
Tasks of Creating a Schematic



Creating a Hierarchical Design

The following figure illustrates the sequence of tasks you perform to create a hierarchical design.

Tasks in Creating a Hierarchical Design



Concept HDL Basics

This section answers the basic questions that are useful when you start working in Concept HDL.

Where can I enter commands?

You can type commands in the console window that appears below the drawing area when you choose *View > Console Window*. If you exit Concept HDL with the console window option enabled, the console window will appear automatically the next time you start Concept HDL.

Command Conventions and Entering Commands

Each menu item has an associated Concept HDL command. To run a command:

- Choose a command from a menu.
- Type a command in the console window, which appears below the drawing area when you choose *View > Console Window*.
- Click a toolbar icon.
- Press the control keys, which are noted next to the frequently used menu commands.
- Draw a stroke pattern.
- Write commands in a script file and run the script.

You can abbreviate Concept HDL commands. Concept HDL recognizes the smallest unique portion of the command name and arguments. Concept HDL commands are not case-sensitive. The Command Reference section of the Concept HDL Online Help shows portions of commands in capital letters to indicate the command abbreviation.

Where are setup options?

Global setup options are located in the Project Manager. You can access Concept HDL setup options both through the Project Manager and through the Tools menu in Concept HDL (*Tools > Options*).

How do I pan drawings?

You can pan a drawing using the mouse, scroll bars, the keyboard, or the *View* menu.

How do I zoom in and out of a drawing?

To zoom into a drawing:

- Choose *View* > *Zoom In*.
- Choose *View* > *Zoom Scale* and enter a scale factor such as 2.
- Choose *View* > *Zoom by Points* and stretch a rectangle around the area you want to zoom into:

  a. Click slightly above and to the left or right of the objects you want to group.

  b. Drag the cursor down diagonally from where you first clicked.

  c. Click again.

To zoom out of a drawing

➤ Choose *View* > *Zoom Out* or *View* > *Zoom Scale* and enter a scale factor such as .5.

To fit a drawing in the screen:

➤ Choose *View* > *Zoom Fit*.

How do I customize Concept HDL?

You can customize toolbars, commands, menus, and keys in Concept HDL using *Tools* > *Customize*.

What commands can I use to edit schematic text?

You can use the following keyboard commands when running the change command (*Text* > *Change)*:

To	Press
Move the cursor backwards	Left Arrow
Move the cursor forward	Right Arrow

To	Press
Move the cursor to the beginning of the line	Home *- or -* Right-click and select *Position at BOL*.
Move the cursor to the end of the line	End *- or -* Right-click and select *Position at EOL*.
Delete the previous character	BackSpace
Delete the next character	Del
Start a text editor	Ctrl + E *- or -* Right-click and select *Editor*.

Are there menu shortcuts?

- Toolbars provide shortcuts to several functions. You can turn on any or all of the toolbars with the *View > Toolbars* menu command.
- Control keys also provide shortcuts to several menu commands. Control-key shortcuts are noted next to the frequently-used menu commands.
- Press predefined function keys (*F1-F12*).
- Standard Windows *Alt* key functions are also available.

How do I browse drawings and components?

The capability to add and edit components used to be contained in a single browser. These are now separate functions.

- Choose *File > Open* to display a file browser from which you select the drawing you want to edit.
- Choose *Component > Add* to display the Component Browser from which you can select components to add to your drawing.

How do I add libraries?

You add libraries using *Tools > Setup* in Project Manager. Within Concept HDL, you can control the available library list and the search order for libraries using *File > View Search Stack*.

How do I add notes?

You can add notes and attach them to the schematic using *Text > Note*.

How do I add parts?

You can add parts using *Component > Add*.

How do I connect parts?

You can connect parts with wires using *Wire > Draw* or *Wire > Route*. *Wire > Draw* lets you manually route around objects while *Wire > Route* automatically routes the wire around objects.

How do I name signals?

You can name signals using *Wire > Signal Name*. You can also create buses by naming signals in the appropriate manner. If you name a wire as `DATA<15..0>`, Concept HDL converts the wire to a 16-bit bus.

How do I add properties?

You can add properties on parts, pins, and signals using *Text > Property*. You can view, add, and modify the visibility of properties using *Text > Attributes*.

How do I add ports?

You can use the ports available in the Standard Library using *Component > Add*.

How do I check my drawing for errors?

- You control settings for error checks in the Concept HDL Setup options accessed through the Tools menu in Concept HDL.
- The *Tools > Check* menu choice or the Check icon in the Standard toolbar lets you run a check.
- You can view error messages and locate them in your design using the Markers control window *(Tools > Markers*). This window also lets you view long, detailed error messages that correspond with the short error messages that are typically displayed.
- Error Status Bar

  Using the status bar in combination with the Markers toolbar, you can view short error messages without the Markers control window. *Tools > Error* controls to navigate the markers file.

How do I save a design?

You can save a design using *File > Save*.

What is Page Locking?

When a user who has write permissions is editing a page in a design, Concept HDL locks the page. If a second user opens the same page for editing, Concept HDL displays a message that the page is locked by the first user and that the second user cannot save any changes made in the page.

Concept HDL creates a lock file called `pagen_csb.lck` in the schematic view when you open a schematic page.

How do I generate a Verilog or VHDL netlist from a schematic?

When you save a design, Concept HDL generates a netlist if you have selected the option for generating a Verilog or VHDL netlist in *Tools > Options > Output*.

How do I add additional pages?

Concept HDL supports multiple page schematics. Choose *File > Edit Page/Symbol > Add New Page* to add a new page to the schematic.

How do I go to a specific page in a design?

1. Choose *File > Edit Page/Symbol > Go To*.

   The *Go To Page/Symbol* dialog box appears.

2. Enter the page number and click *OK*.

To go to a specific page in a hierarchical design, select the *Calculate page number in hierarchy* check box, enter the page number and click *OK*.

Note: If you do not select the *Calculate page number in hierarchy* check box, you can only go to a page within the cell in which the currently open schematic page exists. For example, if the currently open schematic page is `LAPTOP.SCH.1.1`, you can only go to pages within the `LAPTOP` cell.

You can also use the `gotosheet` console command to go to a specific page in a hierarchical design. For more information, see Gotosheet on page 559.

For more information on page numbering in Concept HDL, see Displaying and Working with Schematic Page Numbers on page 358.

How do I plot a design?

You can plot a design using *File > Plot*. On UNIX, you have the option of using the HPF plotting utility also depending on the option you select (Windows plotting or HPF) using *Tools > Options > Plotting*.

What are groups?

When you wish to perform a common edit operation like Copy, Move, or Delete on a collection of objects on the schematic, you can define the collection as a group and carry out the operation using the options available in the Group menu.

What is different about working with groups?

- Functions for creating and working with groups are contained in one group menu.
- A separate toolbar contains the frequently-used group operations.
- Concept HDL makes it easy to set the current group. It clearly shows the group that you are working with at any time by indicating the group name in brackets next to group menu items.

Concept HDL provides a new *Group Contents* dialog box using which you can see the contents of the groups defined in the schematic.

How do I locate parts and wires in a design?

You can locate parts and wires in a design using *Tools > Global Find*. You can also use wildcards on names and narrow down the search using properties and values.

How do I generate a symbol view from a schematic?

You can generate symbol views from schematics using *Tools > Generate View*.

How do I package my design?

You can invoke Packager-XL using the Design Synchronization tool of the Project Manager. You can also use *File > Export Physical* in Concept HDL. For more information on packaging, see *Design Synchronization Online Help*.

How do I backannotate a design?

Backannotation updates the schematic with the layout changes. It annotates your schematic with physical information such as pin numbers and location designators produced by the Design Synchronization process. Choose *Tools > Back Annotate* to specify the file (typically `pstback.dat`) containing the physical information with which you update the schematic.

Caution

Do not run backannotation if any other user who has write permissions is working on the design. Running backannotation when another user is working on the design results in incomplete backannotation.

How do I highlight objects in a design?

To highlight an object in a drawing, choose *Display > Highlight* and click on the object to be highlighted.

You might want to highlight objects in your design for the following reasons:

- To trace a signal on multiple pages of an expanded drawing
- To trace a signal in the drawing hierarchy between expanded drawings
- To correlate the circuit logic to changes you made in the schematic or to navigate the nets between a physical layout and the corresponding schematic between Concept HDL and other system tools.

Choose *Display > Dehighlight* to remove highlighting.

How do I cross-reference a design?

When you view a plot of a schematic, it is often difficult to trace a signal or instances of a part. The Cross Referencer tool traces the signals and parts in a schematic and annotates the location of each one.

On a cross-referenced design, Cross Referencer writes the page number and the location of the part or signal in relation to the page border. These annotations can be found beside each signal and part that has been cross-referenced.

Choose *Tools > CRefer* in Project Manager to cross-reference your design.

How do I archive a design?

You can use the Archiver tool to archive your design. This tool copies over all the libraries that are referenced by your design to the archived area. Archiving lets you work on the design at a location where connectivity to the Libraries server is not available.

To archive your design, choose *Tools > New Archive* in Project Manager**.**

How do I display the online help?

You can display help in several ways:

- Choose *Help > Concept HDL Help*.
- This displays the Help Contents for Concept HDL. Use the Contents tab to navigate the help system, or click the Index tab to search for help using keywords.
- Place the cursor over a menu item and press F1.
- This displays help on the selected command. From the command description, you can display procedures on how to perform the task.

- Click ? in the Standard toolbar and then click an item to be queried.
- Choosing a menu command displays help for the selected command. Clicking in the Concept HDL window displays a brief description of the item.
- Click Help in a dialog box to view information related to that dialog box.
- Type help command_name in the console window.

5

Project Creation and Setup

This section introduces you to the Project Manager tool that you use to create and setup projects and describes the procedures for creating and setting up projects.

Introduction to Project Manager

The Project Manager is the interface to the Cadence board design solution and library management. The Project Manager tool can be used for the following tasks:

- Create design projects or library projects

  Design projects are projects created by designers where the basic aim is creation of designs. Library projects are created by librarians for the sole purpose of library creation and management.

- Set up projects

  You can choose libraries for your project, as well as options such as Physical Part Table files, property files, expansion types, configurations, and view names.

- Import, export, and archive projects
- Launch tools such as Concept HDL, Allegro, SPECCTRAQuest, Library Explorer, Part Developer, and Pad Stack Editor.
- View project settings and libraries in a convenient tree form, and keep track of all tools running in a session.

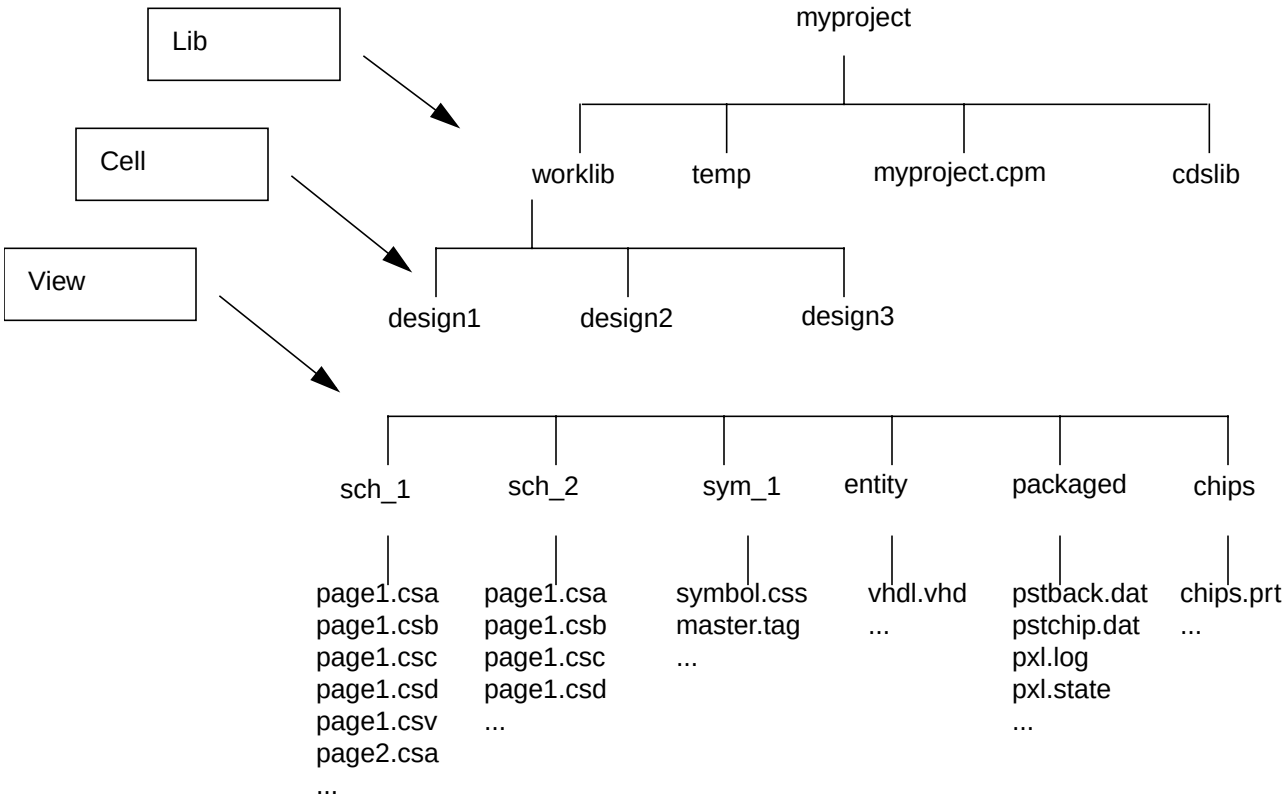
The Project Manager flow can be customized to your requirements. You can add and remove tools from the flow and use custom icons. The flow can also be converted to a toolbar to conserve space on your desktop.

Project Structure

The logical directory structure for designs is `Lib -> Cell -> View -> Files`. Each Lib, Cell, and View is a physical directory.

Lib	It is a directory that contains designs (cells).
Cell	It is the design directory. It contains all the data organized into views for that design. For example, the design *cpu* can have schematic, symbol, chips, packaged, and physical views. The configurations of a design are also stored as views in the design directory.
`View`	It is a directory that contains all the data for a particular unit of a design. For example, all the files for a schematic are in the `sch_n` view (where n is the version number). All the information for a symbol representation of the design is in the `sym_n` view (where n is the version number). The packaged design data is in the packaged view. Configurations, including the four default configurations—`cfg_package`, `cfg_verilog`, `cfg_pic`, and `cfg_vhdl`—are also views under a cell.

Example



Project Files

Project Manager manages all the information about a project—such as its libraries, physical part table files, log files, property files, and setup defaults for tools—through project files.

There are three types of project files:

- Local Project Files
- Site Project File
- Installation Project File

Local Project Files

When you create a new project, the Project Manager creates a project file called <*projectname*>.cpm in the project directory. Each project has one project file. The <*projectname*>.cpm file contains all the setup information that you specified for your project. It has the following:

- The name of the top-level design and the library in which it is located
- The list of project libraries
- The physical part tables selection
- Changes to the default view names
- The name and location of the text editor for editing text files from Cadence tools
- The name and location of the property file
- The name and location of the log file
- The name and location of the application temp directory, which is the directory in which applications such as Concept HDL store temporary files.
- Setup directives for individual tools such as Concept HDL, Packager-XL, and the Project Manager.
- Directives for customizing the Project Manager (a customized Tools menu or customized flows).

The default setup information is maintained in an installation project file (cds.cpm) shipped by Cadence. The defaults in the cds.cpm file apply to all your projects. If you want to change these defaults, create a site project file (site.cpm) for your site.

When you open a project, the Project Manager gets the setup directives you specified for that project from the <*projectname*>.cpm file and the defaults for the others from the site.cpm and cds.cpm files. Your setup directives always have precedence over the site.cpm directives, which in turn have precedence over the cds.cpm directives.

You can view the project settings for a project with the *View* > *Project Settings* command.

Project File (.cpm) Example

```
( Machine generated file created by SPI )
( Last modified was 11:38:31 Thursday, October 09, 1997 )
( NOTE: Do not modify the contents of this file. If this is regenerated by )
( SPI, your modifications will be overwritten. )

START_GLOBAL
use library_ppt 'ON'
design_name 'poa'
design_library 'poa'
library 'poa' 'standard' 'pic' 'poa_lib' 'element'
temp_dir 'temp'
cpm_version '@'
session_name 'ProjectMgr12919'
cdsprop_file ''
ppt './ptf/poa.ppt'
EXCLUDE_PPT
INCLUDE_PPT
END_GLOBAL

START_PKGRXL
state_wins_over_design 'ALL'
END_PKGRXL
```

Site Project File

You create the site project file, called `site.cpm`, in the *<your_inst_dir>*`/share/local/cdssetup/projmgr` directory when you want to specify default setup options for all the projects at your site. The directives in this file have precedence over the installation project file (`cds.cpm`) and the local project file (*<projectname>*`.cpm`) has precedence over the `site.cpm` file. You can override these directives for individual projects by specifying the changes in the local project file (`projectname.cpm`). For more information on creating a site project file, see Creating a Site Project File.

Installation Project File

The installation project file called `cds.cpm` is shipped by Cadence and is in the *<your_install_dir>*`/share/cdssetup/projmgr` directory. The `cds.cpm` file contains the default setup directives for all projects and tools. Project Manager obtains defaults from this file for setup options that are not defined in the `projectname.cpm` or `site.cpm` files. Do not modify this file. If you want to change the defaults for your projects, create a site project file (`site.cpm`).

The setup directives you specify (that is, the directives in the `projectname.cpm` file) always have precedence over the `site.cpm` directives, which in turn have precedence over the `cds.cpm` directives. When you open a project, Project Manager gets the setup directives you

specified for that project from the `projectname.cpm` file and the default values for the other directives from the `site.cpm` file. If the directives are not defined in the `site.cpm` file either, Project Manager obtains the default values from the `cds.cpm` file.

Project Flows

Project Manager is also an HTML browser, and the project flow is defined in a simple HTML document. You can customize the project flow by replacing it with HTML pages that you create for individual projects or for all the projects at your site.

Note: Currently, on UNIX platforms, Project Manager is not an HTML browser. You cannot use HTML files to define custom flows. Instead, use image maps to launch tools.

The standard Cadence Board Design Flow consists of two HTML files, `main.htm` and `home.htm`, which are in the Cadence installation hierarchy at <*`your_install_dir`*>`/share/cdssetup/projmgr/flows`. The `home.htm` file is loaded when no project file is currently open. It has links for opening an existing project or creating a new project. The `main.htm` file is loaded when a project file is opened.

The HTML files contain HREFs. When Project Manager evaluates an HREF, it first looks at its own list of tools to see if the URL matches a tool name. If it does, the tool is launched as a separate process. If no match is found, Project Manager attempts to load the referenced URL. This allows Project Manager to be used as a flow manager, tool launcher, and an internet/intranet browser.

For more information, see the following topics:

- Customizing the default project flow
- HTML code for the default Cadence Board Design Flow
- Example for creating a simple text-based flow
- Example for creating a graphical multiple page flow

Creating a Project

To create a new project in Project Manager, perform the following steps:

1. Choose *File* > *New*. The *New Project Wizard* appears.
2. In the *Name* box, type your project name.
3. In the Location box:

- ❑ Type the complete path of the folder in which you want to create the new project.

- or -

- ❑ Click *Browse*, select a folder in the *Choose Directory* dialog box, and then click *OK*. The *Location* box is filled in with the path of the folder you selected.

Note: If you want to create the project in a new folder, append a name for the new folder to the path (for example: `\cpu`). Project Manager will create the folder.

4. Click *Next*.

The *Project Libraries* dialog box appears with the list of available libraries and project libraries. If you created the project in a new folder or in a folder that does not contain a `cds.lib` file, a `cds.lib` file is automatically created and this file contains a `projectname_lib` entry. You will see the `projectname_lib` entry in the *Project Libraries* list.

5. Select the libraries for your project by placing them in the *Project Libraries* list.

- ❑ To add one library to the *Project Libraries* list, select the library in the *Available Libraries* list and then click *Add*.
- ❑ To add more than one library to the *Project Libraries* list, press *Ctrl* and select the libraries. Then, click *Add*.
- ❑ To add all the libraries in the *Available Libraries* list, click *Add All*.
- ❑ To remove one library from the Project Libraries list, select the library and then click *Remove*.
- ❑ To remove more than one library from the Project Libraries list, press *Ctrl* and select the libraries. Then, click *Remove*.
- ❑ To remove all the libraries from the Project Libraries list, click *Remove All*.

6. Choose the search order for your project libraries. The order in which libraries are listed in the *Project Libraries* list determines their search order.

- ❑ To move a library one level up, select the library and then click *Up*.
- ❑ To move a library one level down, select the library and then click *Down*.

Note: You cannot rearrange the order of the *Available Libraries* list.

7. Click *Next*.

8. In the *Design Name* dialog box, specify the top-level drawing for your design. You can choose an existing design from the project libraries or create a new one in any of the project libraries.

To create a new design:

- In the Library list, select the library in which you want to create the new design.
- In the Design Name box, type a name for the new design.

To select an existing design:

- In the Library list, select the library that contains the design.
- Click Browse, select a design from the *Existing Cell Names* list, and then click *OK*.

9. Click *Next*. The *Finish* dialog box displays your project specifications.

10. Do one of the following:

- To create the project, click *Finish*.
- To change the project name, the project location, the design library, the design name, or project libraries, click *Back* and edit the information you entered in each dialog box. When you finish, click Next until the Finish page appears. Click *Finish* to create the project.

Project Manager displays the default project flow with icons for Concept HDL and Allegro.

Files Created for Your New Project

When you create a new project, Project Manager creates the following:

- A project file (<*projectname*>.cpm)

  The project file contains all the setup information you have specified for the project - the name and location of the top-level design, libraries, view names, physical part tables, and tool setup directives. For more information, see Project Files.

- Four configuration views in the design directory (cell)

  Project Manager creates four configurations for each new project - `cfg_package`, `cfg_verilog`, `cfg_pic`, and `cfg_vhdl`. Each configuration is a directory and contains an `expand.cfg` file.

- An application temp directory (temp)

  Temporary files created by applications such as Concept HDL are placed in the `temp` directory. You can delete the contents of this directory. You can also specify your own application temp directory from Project Setup.

In addition, if you created the project in a new directory or in a directory that does not contain a `cds.lib` file, Project Manager creates the following:

- A `cds.lib` file

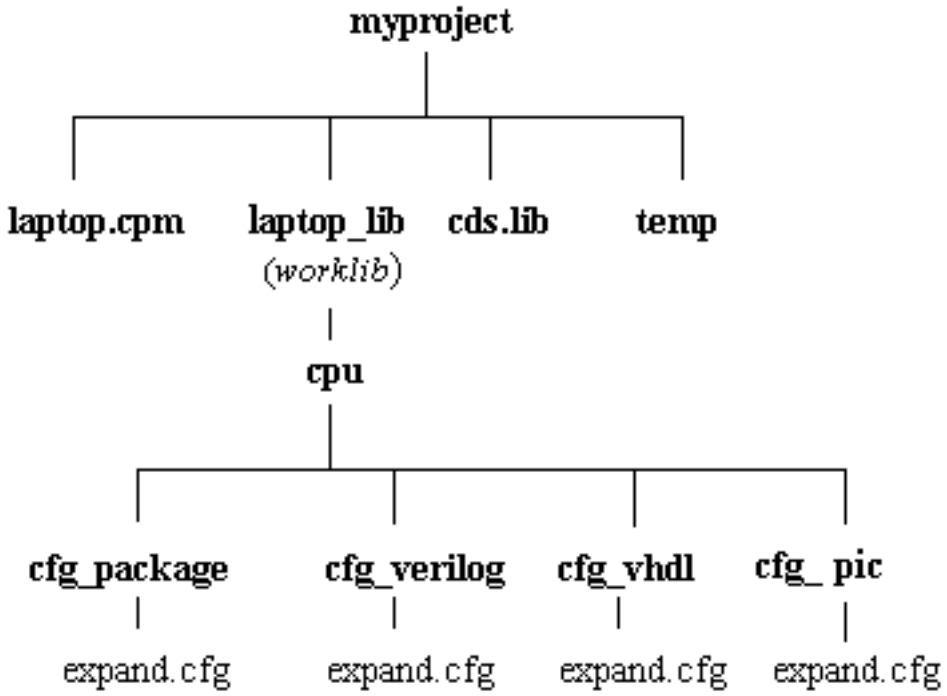
  The `cds.lib` file determines the list of available libraries from which you can choose the project libraries for your project. It contains the logical names of libraries and their physical locations. By default, it includes the path to the installed Cadence libraries. For more information, see the `cds.lib` file.

- A worklib directory.

  The physical name of this directory is `worklib`. You can place your design directories (cells) in this directory. When you use the tools and create designs, non-view log files are added to the project directory, and design data and view-related log files are added to the `worklib` directory.

  Example

  When you create a new project `laptop` in a new directory `myproject` and a design `cpu` in `laptop.lib`, you will have the following file structure:



Creating a Site Project File

You can customize the default settings for all your projects by creating the `site.cpm` file. To create a `site.cpm` file, either use a copy of an existing project file, or create a dummy project and use its project file to define your site settings.

To create a site project file for all the projects at your site,

1. Choose *Tools* > *Setup*.
2. In each tab of the *Project Setup* dialog box, specify the default setup information you want for all projects. For information about the setup options, click the *Help* button in the dialog box.
3. Click *Apply* to save your changes.
4. Close *Project Setup* by clicking *OK*.
5. Choose *File* > *Export*.
6. In the *Export Project* dialog box,
   - Type `site.cpm` in the *File Name* box.
   - In the *Folders* list, select <*your_inst_dir*>/share/local/cdssetup/projmgr, where <*your_inst_dir*> is the directory in which you have installed Cadence tools.
   - Ensure that the *Save File as Type* box displays Project Files (`*.cpm`).
   - Ensure that the *Full Settings* option is not selected.
7. Click *OK* in the *Export Project Setup* dialog box.

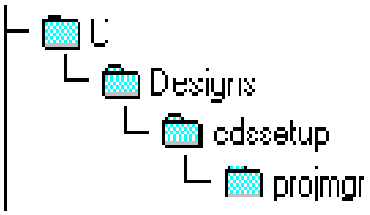
Creating a Custom Site Environment

If you do not place the `site.cpm` file in the <*your_inst_dir*>/share/local/cdssetup/projmgr directory, you must set a `CDS_SITE = location` environment variable that specifies the location of the site project file. The site location must have the following directory structure:

```
cdssetup/projmgr/site.cpm
```

For example, if you want to set your `CDS_SITE = C:\Designs`, you must create the following directory structure and place the `site.cpm` file in the `projmgr` directory:

.



If you have set the CDS_SITE environment variable to another location, such as /hm/common/, you need to ensure that the concepthdl.scr file is installed at /hm/common/cdssetup/concept/so that backannotation from Variant Editor works as required. The site.cpm should be at /hm/common/cdssetup/projmgr/. You also need to copy over any file under /share/cdssetup/ that has been customized to /hm/common/cdssetup/. This will ensure that the customized information is available even when you install a newer version of Cadence PSD software.

If you have any custom Project Manager flows, maintain them at $CDS_SITE/cdssetup/projmgr/flows using the same directory structure as at <your_inst_dir>/share/cdssetup/projmgr/flows/.

If you have customized any of the following files and want the changed version to be available for all projects at your site, copy them into the location recommended below:

Files and Descriptions	**Location**
concepthdl_menu.txt (Concept HDL menus)	Copy from <your_inst_dir>/share/concept/ to $CDS_SITE/concept/.
cref.dat (template options of CRefer)	Copy from <your_inst_dir>/share/cdssetup/creferhdl/ to $CDS_SITE/cdssetup/creferhdl/.
concepthdl_key.txt (Concept HDL shortcut keys)	Copy from <your_inst_dir>/share/concept/ to $CDS_SITE/concept/.
bom.callouts (mechanical parts to be added in the BOM reports)	Copy from <your_inst_dir>/share/cdssetup/ to $CDS_SITE/cdssetup/.
cdsinfo.tag (project-specific information, including the name of the data management system, if any, used in the project)	Copy from <your_inst_dir>/share/cdssetup/ to $CDS_SITE/cdssetup/.

Files and Descriptions	Location
`cdsprop.paf` (information about properties)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`cdsprop.tmf` (information about text macros)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`cjedectype.txt` (compatible JEDEC types in the Variant Editor tool)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`propflow.txt` (the default property flow setup in Packager Setup)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`template.bom` (the default template for BOM reports)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`xmodules.dat` (modules that have to be excluded for cross-referencing and plotting)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`xilfam.dat` (mapping information between a Xilinx family and a specific library and architecture)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`env` (paths to Allegro libraries and other site settings for Allegro)	Copy from `<your_inst_dir>/pcb/text/` to `$CDS_SITE/pcb/text/`.
`allegro.ilinit` (AllegroSKILL initialization file)	Copy from `<your_inst_dir>/pcb/text/` to `$CDS_SITE/pcb/text/`.

Note: The `cdsprop.txt` file need not be copied as you should not be modifying this file.

Setting Up a Project

The Project Setup window displays the current global, physical part table, tools, expansion, and views settings for your project.

You can perform the following tasks from Project Setup:

- Changing the Root Design for a Project
- Creating a New Root Design for a Project
- Editing the cds.lib File
- Selecting Libraries for a Project
- Adding Physical Part Table Files to a Project
- Setting Up Tools
- Specifying the Application Temp Directory
- Selecting a Text Editor
- Selecting a Property File
- Setting Up a Log File
- Selecting an Expansion Type
- Selecting the Configuration for Expansion
- Editing a Configuration
- Creating a New Configuration View
- Selecting Views for the Project

Changing the Root Design for a Project

To change the root design for a project from Project Manager:

1. Open the project for which you want to change the root design.
2. Choose *Tools* > *Setup*. The *Project Setup* window appears.
3. Choose the *Global* tab.
4. In the *Library Name* list, select the library containing the design.
5. In the *Design Name* field, type the name of the design or click *Browse* and select the design from the *Select Cell* list.
6. Click *Apply* to save the changes, or click *OK* to save the changes and exit *Project Setup*.

Note: You can also create a new root design from Project Manager.

Creating a New Root Design for a Project

To create a new root design for a project from Project Manager

1. Open the project in which you want to create a new design.
2. Choose *Tools* > *Setup*. The *Project Setup* window appears.
3. Choose the *Global* tab.
4. In the *Library Name* list, choose the library in which you want to create the new design.

   Note: The *Library Name* list is the list of project libraries.
5. In the *Design Name* field, delete the text and type the new design name. Click *Browse* to see a list of existing cell names for the library you have selected.
6. Click *Apply* to save your changes, or click *OK* to save the changes and exit *Project Setup*.

Editing the cds.lib File

Each project has a `cds.lib` file. Project Manager creates the `cds.lib` file when you create a project in a new folder or in a folder that does not contain a `cds.lib` file. The new `cds.lib` contains:

- A directive to include the installed Cadence libraries. (For example: `INCLUDE <your_install_dir>/share/cdssetup/cds.lib`)
- A define statement that maps the logical project library (`projectname_lib`) to its physical name (`worklib`). (For example: `DEFINE` myproject_lib worklib.)

You can edit the `cds.lib` file and add directives to include any other libraries such as company libraries. You can add libraries to `cds.lib` directly by specifying their logical names and physical locations. Alternatively, you can add a file that contains a list of libraries and their physical locations.

The `cds.lib` file determines the list of available libraries from which you can choose the project libraries for a project.

To edit the cds.lib file

1. Open the project.
2. Choose *Tools* > *Setup*. The *Project Setup* window appears.
3. Choose the *Global* tab.
4. Click the *Edit* button next to the `cds.lib` field. The `cds.lib` file is opened in the default text editor.
5. Edit the `cds.lib` file.

   You can add libraries to the `cds.lib` file directly by specifying their logical names and their physical locations. (For example, DEFINE MYLIB C:/Libraries/IEEE). You can also add files that contain a list of libraries and their locations. (For example, INCLUDE C:/Libraries/company.lib, where `company.lib` contains a list of libraries and their locations.) See Adding Libraries to the cds.lib File.
6. Save the file and exit the text editor.
7. In the confirmation window, click *Yes* to update the library list.
8. Click *Apply* to save your changes, or *OK* to save your changes and exit *Project Setup*.

Adding Libraries to the cds.lib File

To add a library:

➤ Add the following statement to the `cds.lib` file for the project:

```
DEFINE libraryname librarypath
```

where `libraryname` is the logical name for the directory specified in `librarypath`.

The `libraryname` is the name that appears in the list of *Available Libraries* in Project Setup.

Example:

```
DEFINE MYLIB C:/Libraries/IEEE
DEFINE lsttl C:/Libraries/lsttl
```

To add a file containing a list of libraries:

➤ Add one of the following statements to the `cds.lib` file for the project:

```
INCLUDE filename
SOFTINCLUDE filename
```

where *filename* is the name of a file containing a list of libraries and their locations. (*filename* can also be another cds.lib file). Using the INCLUDE statement generates an error message when Cadence tools cannot find this file. Using the SOFTINCLUDE statement generates no error message when Cadence tools cannot find this file.

All the libraries in the cds.lib file will appear in the list of *Available Libraries* in Project Setup.

Example:

```
INCLUDE C:/Libraries/mycompany.lib
```

To remove a library

➤ Add the following statement to the cds.lib file for the project:

```
UNDEFINE libraryname
```

where libraryname is the name of the library you want to remove.

Use this statement when you want to remove some of the libraries defined in a file you included with INCLUDE or SOFTINCLUDE statements.

Selecting Libraries for a Project

1. Open the project.
2. Choose *Tools* **>** *Setup*. The *Project Setup* window appears.
3. Choose the *Global* tab.
4. If you want to view the contents of a library, choose the library and click *View*. A window displaying the contents of the library appears. You cannot make any changes in this window.
5. Modify the *Project Libraries* list under *Library*.
6. To add one library, select the library in the *Available Libraries* list and click *Add*.
7. To add all the libraries in the *Available Libraries* list, click *Add All*.
8. To remove one library, select the library in the *Project Libraries* list and click *Remove*.
9. To remove all the libraries in the *Project Libraries* list, click *Remove All*.
10. Choose the search order for the project libraries. The order in which the libraries are listed in the *Project Libraries* list determines their search order.

11. To move a library one level up, select the library and then click *Up*.

12. To move a library one level down, select the library and then click *Down*.

13. Click *Apply* to save your changes, or *OK* to save the changes and exit *Project Setup*.

Available Libraries and Project Libraries

Available Libraries

They are the libraries available to you for any project. These are determined by the directives in the `cds.lib` file. The Cadence-installed libraries are included in the `cds.lib` file as the default libraries. You can edit the `cds.lib` file to add other libraries to the list of available libraries.

Project Libraries

They are the libraries you select for your project from the list of available libraries. You can select project libraries when you create a project or any other time using the Setup tool. If you create a project in a new folder or in a folder that does not have a `cds.lib` file, a `projectname_lib` file is also created and placed in the *Project Libraries* list.

You can modify the *Project Libraries* list from Project Manager.

Adding Physical Part Table Files to a Project

The Physical Part Table (`.ptf`) file contains the data you need to add or modify the physical properties of a symbol. The `.ptf` files can be located at the cell level under the Part Table view, or in any other directory. Cell-level `.ptf` files contain information about the primitives for that cell.

To access the information contained in the Physical Part Table files, you must include them in your project. When you include cell-level Physical Part Tables, all the `.ptf` files in the Part Table view of that cell are included. You can also include other `.ptf` files by specifying their location.

You can include cell-level `.ptf` files and other `.ptf` files in the same project. If you have a cell-level `.ptf` file, then Packager-XL does not read it if the INCLUDE_PPT directive is set. To include the cell-level `.ptf` file, you will have to add it in the INCLUDE_PPT directive.

To add physical part tables to your project, you can add either `.ptf` files directly or directories that contain `.ptf` files. For example, if the `lsttl` directory contains the `lsttl.ptf` file, you can add either the complete path to the `lsttl.ptf` file or just the path to the `lsttl`

directory. When you add a directory, all the `.ptf` files in that directory are added to the project. You can then exclude some of the `.ptf` files if you do not want them in the project.

The following steps are required to add Physical Part Table files to a project:

1. Open the project.
2. Choose *Tools > Setup*. The *Project Setup* window appears.
3. Select the *Part Table* tab.
4. To add cell-level `.ptf` files to your project, select the *Use Cell-Level Physical Part Table Files* check box. All the `.ptf` files contained in the Part Table view of the cells will be read by Packager-XL.
5. To add other Physical Part Table files,
   a. Under *Physical Part Table Files*, click *Add*. The *Add Physical Part Table* dialog box appears.
   b. Type the name and the path of the `.ptf` file or the directory containing the `.ptf` files. To add more than one path, separate each path with a space.
   —or—
   To add a file, click *File...* and select the `.ptf` file in the *Choose Physical Part Table Files* dialog box. (To select more than one file, select the first one, then press *CTRL* and select the others.) To add a directory, click *Directory...* and select a directory in the *Choose Directory* dialog box.
   c. Click *OK*.
6. To exclude any unwanted `.ptf` files contained in the directories you have added, do one of the following:
   - Under *Exclude Physical Part Table Files*, click *Add* and enter the name and path of the `.ptf` file you want to exclude from your project. Repeat this step for all the files you want to exclude.
   - Under *Include PTFs*, click *Add* and enter the name and path of the `.ptf` file you want to include in your project. Repeat this step for all the files you want to include.
   - To remove a Physical Part Table file or directory, select the file and click *Remove*.
   - Select the *Merge Physical Part Table Files* check box to merge the information in all included Physical Part Table files.
   - Select the *Perform Case Sensitive Row Match* check box to perform a case-sensitive match of key properties for a part in the physical part table files.
7. Click *Apply* to save your changes, or *OK* to save your changes and exit *Project Setup*.

8. You can include cell-level `.ptf` files and other `.ptf` files in the same project – Packager-XL reads the contents of each.

Setting Up Tools

The Tools tab in the Project Setup window allows you to select the setup options for Allegro, Concept HDL, Project Manager, Packager-XL, Programmable IC, Simulation, and Mixed Signal simulation. You can specify setup directives for these tools from Project Manager or directly from the tools. The Tools tab also displays the default settings for the property file, the text editor, the project log file, and the temp directory.

In this tab, you can:

- Specify the setup directives for Allegro, Concept HDL, Project Manager, Packager-XL, Programmable IC, Simulation, and Mixed Signal Simulation. Simulation Interface provides a simulation environment to simulate your schematics from Concept HDL with Verilog or Leapfrog. You can specify setup directives for Concept HDL and Packager-XL directly from the tools or from Project Manager.
- Select a default text editor.
- Specify an application temp directory.
- Select a property file.
- Set up a log file.

Note: You can specify the setup directives for Concept HDL and Packager-XL directly from the tools or from Project Manager.

Specifying the Application Temp Directory

The Application Temp directory is the directory in which applications such as Concept HDL store temporary files. You can delete the contents of this directory.

An Application Temp directory, `temp`, is created automatically when you create a new project. However, you can specify any directory as the Application Temp directory for a project.

To specify an Application Temp directory:

1. Choose *Tools* > *Setup*. The *Project Setup* window appears.
2. Select the *Tools* tab.

3. In the *Temp Directory* field, type the full path to the *temp* folder, or click *Browse* and use the file browser to select the location of the *temp* folder.
4. Click *Apply* to save your changes, or click *OK* to save your changes and exit Project Setup.

Selecting a Text Editor

For each project, you can select a text editor as the default text editor for Cadence tools. When you view or edit any text file from a Cadence tool, it will be displayed in the text editor you have specified. The default editor is WordPad on Windows NT and vi on UNIX.

To select a text editor:

1. Choose *Tools* > *Setup*. The *Project Setup* window appears.
2. Select the *Tools* tab.
3. In the *Default Text Editor Path* field, type the full path to the text editor you want to use, or click *Browse* and use the file browser to select the text editor.
4. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

Selecting a Property File

The property file for a project contains directives that control how properties are handled during expansion. It specifies whether a property is inherited by other objects, whether it is a parameter, what objects it can be attached to, and whether it is passed to the destination tool.

Cadence provides a default property file called `cdsprop.paf`, which is located in the <*`your_install_dir`*>/`share/cdssetup` directory. Do not modify this file. You can use your own property file by specifying its path in *Project Setup*.

To select a property file:

1. Choose *Tools* > *Setup*. The *Project Setup* window appears.
2. Select the *Tools* tab.
3. In the *Property File* field, type the full path of the property file you want to use, or click *Browse* and use the file browser to select the file.
4. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

Example `cdsprop.paf` file

```
FILE_TYPE=ATTRIBUTES;
{ Default attributes for properties.
  Attributes for user designed properties should be added to the user's
property attribute file. This file should not be modified. }
ALLOW_CONNECT:inherit(signal), permit(pin, body,signal);
AUTO_GEN: inherit(), permit(body);
BIDIRECTIONAL:inherit(), permit(pin);
BODY_NAME:     inherit(), permit(body);
CHIP_DELAY:    inherit(pin), permit(pin, signal);
CLOCK_DELAY:inherit(pin), permit(pin, signal);
COMMENT_BODY:filter;
COUPLED:       inherit(), permit(body);
DELAY:              inherit(), parameter, permit(body);
DIR:                inherit(), permit(body), case_sensitive;
EVAL:               inherit(pin), permit(pin, signal);
EXPR:               filter;
FALL:               inherit(), parameter, permit(body);
GROUP:         inherit(body), permit(body);
HAS_FIXED_SIZE:inherit(), permit(body);
HIGH:               inherit(), permit(body);
INPUT_LOAD:    inherit(), permit(pin);
IO_NET:             inherit(signal), permit(signal);
LAST_MODIFIED:inherit(), filter;
LOCATION:      inherit(body), permit(body);
LOCATION_CLASS:inherit(body), permit(body);
LOW:                inherit(), permit(body);
MODEL:              inherit(), permit(body);
```

Setting Up a Log File

A log file for a project tracks information such as the date and time of any activity, the tools launched from the project, the user's name, and MPS sessions and hosts.

If you want to maintain a log file for the project, you must select the option in Project Setup. A log file will not be generated by default.

To set up a log for a project:

1. Choose *Tools* > *Setup*. The *Project Setup* window appears.
2. Select the *Tools* tab.
3. In the *Project Log File* field, type a name for the log file. The file will be created in the project directory. To specify an existing file, click *Browse* and use the file browser to select it.
4. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

Selecting an Expansion Type

The expansion type and configuration you select in Project Setup determines the current configuration for Concept HDL and Hierarchy Editor. If you change the expansion type in Project Setup, the current configuration for Concept HDL and Hierarchy Editor changes. The default expansion type is Physical Layout.

To select the expansion type for your design:

1. Choose *Tools > Setup*. The Project Setup window appears.
2. Select the *Expansion* tab.
3. Do one of the following:
   - Select the *Physical Layout* option to expand your design for Allegro and other back-end tools.
   - Select the *Verilog Simulation* option to expand your design for simulation with Verilog-XL and other Verilog-based simulators.
   - Select the *VHDL Simulation* option to expand your design for simulation with Leapfrog and other VHDL-based simulators.
   - Select the *PIC Configuration* option to expand your design for simulation with Programmable IC such as Verilog-XL.
   - Select the *Mixed Signal* option to expand your design for mixed-signal simulation.
4. In the *View* field next to the expansion type you selected, click *Browse*. The *Select View* dialog box appears. Select the configuration you want to expand and click *OK*.
5. If you want to view or edit the configuration, click *Edit*. The configuration is opened in the Cadence Hierarchy Editor. Edit the configuration, save it with the *File > Save* command, and exit the Cadence Hierarchy Editor.
6. Click *Apply* to save your changes, or click *OK* to save your changes and exit Project Setup.

Note: When you package a design, Packager-XL always uses the Physical Layout expansion, irrespective of the expansion type you select in Project Setup.

Selecting the Configuration for Expansion

When you create a new project, the default configurations for each expansion type are created automatically. These are:

- `cfg_package` for Physical Layout
- `cfg_verilog` for Verilog Simulation
- `cfg_vhdl` for VHDL Simulation
- `cfg_pic` for PIC Simulation
- `cfg_mixed` for Mixed Signal Simulation

You can select a different configuration for each expansion type.

To select the configuration for each expansion type:

1. Choose *Tools* > *Setup*. The *Project Setup* window appears.
2. Select the *Expansion* tab.
3. Click the *Browse* button next to expansion type. The *Select View* dialog box appears. Select the configuration you want to use and click *OK*.
4. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

Note: You can also create a new configuration view from Project Manager.

Editing a Configuration

You edit a configuration with the Hierarchy Editor, a graphical tool for creating and editing configurations.

To edit a configuration:

1. Choose *Tools* > *Setup*. The Project Setup window appears.
2. Select the *Expansion* tab.
3. In the View field next to an expansion type (Physical Layout, Verilog Simulation, VHDL Simulation, PIC Configuration and Mixed Signal), click *Browse* and select the configuration you want to edit.
4. Click *Edit*. The configuration you specified in the *View* field is opened in the Cadence Hierarchy Editor.
5. Make the required changes in the Hierarchy Editor, save the changes, and exit the Hierarchy Editor.
6. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

Creating a New Configuration View

1. Choose *Tools* > *Setup*. The Project Setup window appears.
2. Select the *Expansion* tab.
3. In the *View* field next to the *Expansion Type* option you have chosen, delete the existing view name, and type the name for the new view.
4. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

Project Manager creates the new view as well as an `expand.cfg` file in the view.

Selecting Views for the Project

Views are created when you work with your designs. When you package a design, a `packaged` view is created and all Packager-XL output files and log files are stored in it. The `chips.prt` file is placed in the `chips` view, cell-level physical part tables in the `part_table` view, and PCB data in the `physical` view.

Similarly, the root design views for Board Design, PIC Design, Verilog Simulation and VHDL Simulation will be used for expanding your design for physical layout, PIC simulation, Verilog simulation, and VHDL simulation, respectively.

Project Manager assigns default view names.These are:

Type of View	View Name
Packaged	packaged
Chips	chips
Part Table	part_table
Physical	physical
Constraints	constraints
Board Design	sch_1
Programmable IC Design	sch_1
Verilog Simulation	sim_sch_1
VHDL Simulation	sim_sch_1

You can change the view names for each project.

Changing View Names

You can create a new view name or select an existing view name. Open the project for which you want to change view names.

1. Choose *Tools* > *Setup*. The *Project Setup* window appears.
2. Select the *Views* tab.
   - ❑ To change the view name, click on the drop-down list to select from an existing view name.
   - ❑ To create a new view name, type the new view name.
3. Click *Apply* to save your changes, or *OK* to save your changes and exit Project Setup.

6

The Concept HDL Editing Environment

Concept HDL enables you to make changes in the editing environment according to your preferences. The following topics are described below:

- Setting Up Defaults on page 89
- Basic Editing Tasks on page 91
- Displaying Information on page 98
- Basic Navigation in Concept HDL on page 106
- Running Commands with Strokes on page 112
- Modes in Concept HDL on page 113

Setting Up Defaults

This section covers the following information:

- Setting Up Concept HDL Editor Options
- Defining a Default Text Editor

Setting Up Concept HDL Editor Options

1. Choose *Tools > Options*.
2. Click a tab to display the setup options you want to view.
3. For example, click *Grid* to display the grid options.
4. Click *OK*.

Enabling the Pre-Select Menu Use Model

The post-select menu use model is the default model. You can set the preferred menu use model to the pre-select use model.

1. Choose *Tools > Options > General*.
2. Select the *Enable Pre Select Mode* check box.
3. Click *Apply*.
4. Click *OK*.

Now, for operations on schematics, select the object first, and then the menu option for the operation you want to perform.

For example, to rotate a component using the pre select model, first click on the component and then choose the menu option *Edit > Rotate*.

1. Select the component.
2. Choose *Edit > Rotate*.

Note: Concept HDL supports the pre-select mode only through menus; the pre-select mode is not supported through Concept HDL commands. For example, in this mode if you first select an object, and then type *delete* in the console window, the object is not deleted.

Setting Automatic Page Borders

To automatically set a page border for a new schematic:

1. Choose *Tools > Options > General*.
2. In the *Page Border* section:
   a. Specify the name of the page border you want to add in the *Symbol* field.
   b. Specify the version of the page border in the *Version* field. By default, the version is 1.
3. Click *Apply*.
4. Click *OK*.

Defining a Default Text Editor

If you open a text file in Concept HDL, the default text editor is used to display and edit the file.

To set the default text editor for a project:

1. Open Project Manager.
2. Click on the *Setup* icon.

   The Project Setup dialog box appears.
3. Select the *Tools* tab.
4. Specify the Default Text Editor Path.

   The default text editor is vi on UNIX and write on Windows NT. These default settings are read by Project Manager from <*your_install_dir*>/share/cdssetup/projmgr/cds.cpm.
5. Click *Apply*.

Basic Editing Tasks

This section covers the steps to be performed in Concept HDL to complete basic editing tasks, such as undo, copy, paste, delete, and so on.

- Undoing an Operation on page 91
- Moving Objects on page 92
- Copying Objects on page 94
- Deleting Objects on page 95
- Changing the Color of Objects on page 96
- Drawing an Arc on page 96
- Drawing a Circle on page 96
- Splitting Overlaid Objects on page 97
- Displaying the Console Window on page 97
- Editing Text in Dialog Boxes and the Console Window on page 98

Undoing an Operation

To undo an operation:

1. Choose *Edit* > *Undo*.

2. Continue choosing *Edit* > *Undo* to back out of operations progressively.
3. You can reverse an undo operation by choosing *Edit* > *Redo*.
4. If you have moved objects between drawings and want to undo the operation, you must choose *Edit* > *Undo* once in each of the drawings.

Note: You can also run the *undo* command using the following stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes.

Reversing an Undo

To redo an operation:

1. Choose *Edit* > *Redo*.
2. Continue choosing *Edit* > *Redo* to reverse undo operations progressively.
3. You can reverse the redo operation by choosing *Edit* > *Undo*.

Moving Objects

To move text, wires, or an unwired component

1. Select the object.
2. Move the object to a new location and click again.

OR

1. Choose *Edit* > *Move*.
2. Click on the object.

   Note: When moving a wire, select the middle of the wire. Selecting the wire at the end stretches it.
3. Move the object to a new location and click again.

To move a wired component

1. Ensure that *Auto Route on Move* is checked in *Tools* > *Options* for Graphics.

2. Choose *Edit > Move*.
3. Click on the wire nearest to the component you want to move.

   The following three cases can occur depending on where you click the mouse button and which button you click:

   Click LMB near the open edge of the wire.

   Only the wire moves, the component attached to it does not move.

   Click LMB on the wire segment near the component pin.

   Both the component and the wire attached to it move.

   Click RMB and select the context menu item Change Attachment.

   There are three possible states of the move operation:

   - The first time selection of the *Change Attachment* menu item moves the component along with all the wires connected to it. The attached wires are directly routed.
   - The second time selection of the *Change Attachment* menu item disconnects the wire segment from the pin of the component and moves the wire segment.
   - The third time selection of the *Change Attachment* menu item moves just the wire segment while keeping it connected to the component pin.

Note: You can also run the move command using the following stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes.

To move multiple objects

1. Hold down the left mouse button and drag the mouse to select multiple objects, or use *Ctrl+click* or *SHIFT+click* to select multiple objects.

   To exclude components, properties or wires from the selected objects, click the right mouse button and choose *Exclude* to exclude components, properties or wires from the selected objects.
2. Click on one of the selected objects.
3. Move the objects to a new location and click again.

Copying Objects

To copy an object:

1. Choose *Edit > Copy*.

   To copy an object with its properties, right-click and choose *All* from the pop-up menu.

2. Click an object.

   The object is attached to the cursor. You can place the object on the drawing.

3. To place several copies of the object without specifying the object again, right-click and choose *Retain Selection* from the pop-up menu.

4. Click in the same drawing or in another window to place the copies.

5. If you chose *Retain Selection* from the pop-up menu and you are done placing copies but wish to remain in the *Edit > Copy* mode, choose *Terminate Selection* from the pop-up menu. To exit the *Edit > Copy* entirely, choose *Done* from the pop-up menu.

Note: You can also run the copy command using the following stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes.

To copy an object and its properties:

1. Choose *Edit > Copy All*.

2. Click an object.

   The object is attached to the cursor. You can place the object on the drawing.

3. To place several copies of the object without specifying the object again, right-click and choose *Retain Selection* from the pop-up menu.

4. Click in the same drawing or in another window to place the copies.

5. If you chose *Retain Selection* from the pop-up menu and you have placed copies but wish to remain in the *Edit > Copy All* mode, choose *Terminate Selection* from the pop-up menu. To exit *Edit > Copy All* entirely, choose *Done* from the pop-up menu.

Note: You can also copy properties when you choose *Edit > Copy* or *Edit > Array*, right-click, and choose *All* from the pop-up menu.

To make multiple copies of an object:

1. Choose *Edit > Array*.
2. Type the number of copies you want to make in the *Array Size* box, and click *OK*.
3. To copy an object with its properties, right-click and choose *All* from the pop-up menu.
4. Click in the same drawing or in another window to place the copies.

   Objects copied in the array are offset from each other by the same distance as the first object in the array from the original.
5. To place another copy of the array without specifying the array again, right-click and choose *Retain Selection* from the pop-up menu.
6. Click in the same drawing or in another window to place the array.

   A copy of the selected object remains attached to the cursor. You can place the copy in several unrelated places on the drawing.
7. If you chose *Retain Selection* from the pop-up menu and you have placed the array but wish to remain in *Edit > Array* mode, choose *Terminate Selection* from the pop-up menu. To exit *Edit > Array* entirely, choose *Done*.

To copy multiple objects

1. Hold down the left mouse button and drag the mouse to select multiple objects, or use *Ctrl+click* or *SHIFT+click* to select multiple objects.

   To exclude components, properties or wires from the selected objects, click the right mouse button and choose *Exclude* to exclude components, properties or wires from the selected objects.
2. Click the right mouse button and choose *Copy*, *Copy All* or *Array*.

   The objects are attached to the cursor. You can place the objects on the drawing.

Deleting Objects

To delete an object:

1. Choose *Edit > Delete*.
2. Click on the objects you want to delete.
3. To reverse a deletion, choose *Edit > Undo*.

Note: You can also run the `delete` command using the following stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes.

To delete multiple objects

1. Hold down the left mouse button and drag the mouse to select multiple objects, or use *Ctrl+click* or *SHIFT+click* to select multiple objects.

   To exclude components, properties or wires from the selected objects, click the right mouse button and choose *Exclude* to exclude components, properties or wires from the selected objects.

2. Click the right mouse button and choose *Delete* to delete the objects.

Changing the Color of Objects

To change the color of an object:

1. Choose *Edit > Color*.
2. Click on the objects whose color you want to change.

Drawing an Arc

To draw an arc:

1. Choose *Edit > Arc*.
2. First click in the schematic, then move the cursor to approximately the diameter of the arc and click again.

   These first two points define the endpoints of the arc.

3. Click a third time between the two points.

The curvature of the arc is defined by how close to the first two points you click. You can also draw a circle from an arc.

Drawing a Circle

To draw a circle:

1. Choose *Edit > Circle*.

2. Click in the schematic.

   This defines the center of the circle.

3. Size the circle by dragging the cursor away from the center.

To draw a circle from an arc:

1. Choose *Edit* > *Arc*.
2. Click in the schematic, and move the cursor to mark the desired diameter of the circle and click again.
3. Right-click and choose *Done* from the pop-up menu.

Splitting Overlaid Objects

1. Choose *Edit* > *Split*.
2. Click the overlaid objects.
3. Click a clear location nearby to place the overlaid object(s).

UPPER CASE	Abbreviated entry for the command
Bold	Required input
[ ]	Optional input
Italic	Variable
...	Continuous until you exit the command mode
;	Exits the command mode
click	Mouse clicks

Displaying the Console Window

➤ Choose *View* > *Console Window*.

The console window appears. The next time you display the *View* menu, you'll see a check appears next to *Console Window*.

Editing Text in Dialog Boxes and the Console Window

To	Press
Move the cursor left, right, up, or down	Arrow keys
Move the cursor to the beginning of line	Home
Move the cursor to the end of line	End
Select text (or extend selection) to the left, right, up, or down	Shift + Arrow keys
Select text (or extend selection) to the end of line	Shift + End
Select text (or extend selection) to the beginning of line	Shift + Home
Delete the previous character	Backspace
Delete the next character	Delete
Delete selected text	Delete
Copy selected text	Ctrl + Insert - or - Ctrl + C - or - Copy (*UNIX only*)
Cut selected text	Ctrl + X - or - Cut (*UNIX only*)
Paste	Shift + Insert - or - Ctrl + V - or - Paste (*UNIX only*)
Undo the last cut or paste	Alt + Backspace

Note: On Windows NT, you can also use commands from the popup menu to edit text in dialog boxes. These commands are not available on UNIX platforms.

Displaying Information

This section covers the following information:

■ Displaying Schematic Information on page 99

- Displaying Toolbars and Other Parts of the Concept HDL Window on page 103
- Highlighting Objects on page 105
- Turning Off Highlighting on page 105
- Opening the Markers Control Window on page 105
- Displaying the Markers Toolbar on page 106
- Displaying the Error Status Bar on page 106

Displaying Schematic Information

Concept HDL lets you highlight selected objects in drawings, between drawings, and between Concept HDL and other tools.

Concept HDL also displays information about these items:

Attachments	Modified (drawings)
Color	Nets
Component	Origins (objects)
Connections(wires)	Pins (locations)
Coordinates	Pin Names
Directory (current)	Properties
Distance (point to point)	Return
History (drawings)	Text Size
Keys (assignments)	

Displaying Attachments Between Properties and Objects

➤ Choose *Display > Attachments*.

Attachments display in red.

Note: Choose *Window > Refresh* to clear your selections or any Concept HDL display information.

Displaying Component Information

1. Display the console window.
2. Choose *Display > Component*.

   Component information is displayed in the console window.

Displaying the Color of Objects

1. Display the console window.
2. Choose *Display > Color*.

   Component information is displayed in the console window.

Displaying Wire Connections

➤ Choose *Display > Connections*.

An asterisk appears at each wire connection.

Note: Choose *Window > Refresh* to clear your selections or any Concept HDL display information.

Displaying Coordinates

1. Display the console window.
2. Choose *Display > Coordinates*.
3. Click in the schematic.

   The x.y location of a point is displayed in the console window.

Note: The point you specify will clear with your next menu selection.

Displaying the Current Directory

1. Display the console window.
2. Choose *Display > Directory*.

   The current directory is displayed in the console window.

Displaying the Distance Between Points

1. Display the console window.
2. Choose *Display* > *Distance*.
3. Click at one point in the schematic and then at a second point.

   The distance between them is displayed in the console window.

Note: The points you specify will clear with your next menu selection.

Displaying the Drawings Read Into Concept HDL in the Current Session

1. Display the console window.
2. Choose *Display* > *History*.

   Drawing names are listed in the console window.

Displaying Function Key Assignments

➤ Choose *Display* > *Keys*.

Function key assignments are listed in a message box.

Displaying Modified Drawings

1. Display the console window.
2. Choose *Display* > *Modified*.

   Drawings that were modified but not saved are listed in the console window.

Displaying Nets

1. Choose *Display* > *Net*.
2. Click a wire (net).

   The selected net is displayed in red.

Note: Choose *Window* > *Refresh* to clear your selections or any Concept HDL display information.

Displaying Origins of Objects

➤ Choose *Display* > *Origins*.

An asterisk appears at the origin of all objects in the schematic.

Note: Choose *Window* > *Refresh* to clear your selections or any Concept HDL display information.

Displaying Pins

➤ Choose *Display* > *Pins*.

An asterisk appears at all pin locations in the schematic.

Note: Choose *Window* > *Refresh* to clear your selections or any Concept HDL display information.

Displaying Pin Names

1. Zoom in on the component whose pin names you want to display.
2. Choose *Display* > *Pin Names*.
3. Click the component.

   Pin names appear next to each of the pins.

Note: Choose *Window* > *Refresh* to clear your selections or any Concept HDL display information.

Displaying Property Information

➤ Choose *Display* > *Properties*.

Property names and their values appear for each property in the schematic.

Note: Choose *Window* > *Refresh* to clear your selections or any Concept HDL display information.

Displaying the Name of the Previous Drawing

1. Display the console window.
2. Choose *Display* > *Return*.

The names of drawings that were edited in the current window are listed in the console window.

Displaying Text Size

1. Display the console window.
2. Choose *Display > Text Size*.
3. Select some text.

   The size of the selected text is displayed in the console window.

Note: Choose *Window > Refresh* to clear your selections or any Concept HDL display information.

Displaying Pages in a Multipage Drawing

To view the next page of a drawing

➤ Choose *File > Edit Page > Next*.

To view the previous page of a drawing

➤ Choose *File > Edit Page > Previous*.

To view a specific page of a drawing

➤ Choose *File > Edit Page > Go To*.

To display the previous window

➤ Choose *View > Previous View*.

Concept HDL displays the drawing as it appeared before you changed the view.

Displaying Toolbars and Other Parts of the Concept HDL Window

To display toolbars:

1. Choose *View > Toolbars.*

2. Click the check boxes next to the toolbar you want to display:
   - ❑ Standard
   - ❑ Add
   - ❑ Block
   - ❑ Markers
   - ❑ Edit
   - ❑ Color
   - ❑ Analog
   - ❑ Passive
   - ❑ Source
   - ❑ Linear
   - ❑ Discrete
   - ❑ Misc

   Concept HDL places a check in the box next to the toolbar you want to display. If a toolbar is already checked (on) and you click the check box next to the item, Concept HDL removes the check, turning off that toolbar.
3. Click *OK*.

To display a grid, status bars, or the console window

➤ Choose *View* and then choose one of the items you want to display:
   - ❑ Grid
   - ❑ Status Bar
   - ❑ Error Status Bar
   - ❑ Console Window

   When you choose one of these items, Concept HDL places a check next to it, turning it on. If an item is already checked (on), Concept HDL turns it off and removes the check.

Highlighting Objects

To highlight an object:

1. Choose *Display* > *Highlight*.
2. Click an object.

To highlight multiple objects

1. Hold down the left mouse button and drag the mouse to select multiple objects, or use *Ctrl+click* or *SHIFT+click* to select multiple objects.

   To exclude components, properties or wires from the selected objects, click the right mouse button and choose *Exclude* to exclude components, properties or wires from the selected objects.
2. Click the right mouse button and choose *Highlight* to highlight the objects.

Turning Off Highlighting

To dehighlight an object:

1. Choose *Display* > *Dehighlight*.
2. Click the pin, wire (net), or component that you want to dehighlight.

To dehighlight multiple objects

1. Hold down the left mouse button and drag the mouse to select multiple objects, or use *Ctrl+click* or *SHIFT+click* to select multiple objects.

   To exclude components, properties or wires from the selected objects, click the right mouse button and choose *Exclude* to exclude components, properties or wires from the selected objects.
2. Click the right mouse button and choose *Dehighlight* to dehighlight the objects.

Opening the Markers Control Window

To view markers:

➤ Choose *Tools* > *Markers*.

or

Click in the *Markers* toolbar.

Displaying the Markers Toolbar

1. Choose *View* > *Toolbars*.

   The *Toolbars* dialog box is displayed.
2. Check *Markers*.
3. Click *OK*.

Displaying the Error Status Bar

➤ Choose *View* > *Error Status Bar*.

or

Click in the *Markers* toolbar.

To enlarge the drawing:

1. Choose *View* > *Zoom by Points*, or click the *Zoom Points* icon in the standard toolbar.
2. Draw a rectangle around the component to be sectioned.

Alternatively, you can choose *View* > *Zoom In*, or click the *Zoom In* icon to zoom in incrementally.

Basic Navigation in Concept HDL

This section covers the following information:

- Panning the Drawing on page 107
- Zooming In and Out of the Drawing on page 107
- Navigating the Drawing Hierarchy on page 108
- Moving a Window on page 109
- Resizing a Window on page 109
- Closing a Window on page 109
- How do I navigate a design? on page 109
- Finding Nets and Cells in Your Design on page 110

■ Navigating Nets in Your Design on page 111

■ Exiting Concept HDL on page 112

Panning the Drawing

To pan using the mouse:

1. Press and hold the right mouse button or press *SHIFT* and hold the right mouse button.
2. Move the mouse to view portions of the drawing.

Note: Enabling the *Window Autopan* option causes the window to move over the drawing. Turning off *Window Autopan* causes the drawing to move inside the window. (See the *General* tab under *Tools* > *Options*).

To pan using scroll bars:

➤ Click the slider in either the vertical or horizontal scroll bar and drag it.

To pan using the keyboard:

1. Press and hold *Ctrl*.
2. Press any arrow key (up, down, left, right).

To pan using the View menu:

➤ Choose one of these:

View > *Pan Up*

View > *Pan Down*

View > *Pan Left*

View > *Pan Right*

Zooming In and Out of the Drawing

To zoom into the drawing:

■ Choose *View* > *Zoom In*

- Choose *View > Zoom Scale* and enter a scale factor, such as 2.
- Choose *View > Zoom by Points* and stretch a rectangle around the area you want to zoom in.

1. Click slightly above and to the left or right of the objects.
2. Move the cursor down diagonally from where you first clicked.
3. Click again.

To zoom out of the drawing:

➤ Choose *View > Zoom Out* or *View > Zoom Scale* and enter a scale factor, such as .5.

To fit the drawing in the screen:

➤ Choose *View > Zoom Fit*.

Navigating the Drawing Hierarchy

To view a block diagram from the top-level schematic:

1. Choose *File > Edit Hierarchy > Descend*.
2. Click a block in the schematic.
3. Click *OK* in the message box that appears.

   The block diagram is displayed.
4. Continue descending the drawing hierarchy by repeating steps 1 and 2.

To ascend the drawing hierarchy from a lower level block diagram:

1. Choose *File > Edit Hierarchy > Ascend*.
2. Click *OK* in the message box that appears.
3. Continue ascending the drawing hierarchy by repeating steps 1 and 2.

To return to the previous drawing:

1. Choose *File > Return*.

Note: You can view the list of the drawings that Concept HDL will return to and the order in which the drawings will be accessed by choosing *Display > Return*.

Descending into the Drawing:

1. Choose *File > Descend*.
2. Click the block or component whose logic you want to view.
3. Choose *File > Return* or *File > Ascend* to descend to the previous drawing.

Moving a Window

1. Place the cursor in the title bar of the window.
2. Press and hold the left arrow key.
3. Slide the window to a new location.

Resizing a Window

1. Place the cursor in a corner of the window.

   The cursor changes to a diagonal two-headed arrow.
2. Press and hold the left arrow key.
3. Stretch or reduce the window to a new size.

Closing a Window

➤ Choose *File > Close*.

Closing a window does not save the design. Concept HDL saves your design only if you choose *File > Save* or when you exit Concept HDL.

How do I navigate a design?

- Choosing either *File > Edit Hierarchy > Descend* or *File > Edit Hierarchy > Ascend* lets you navigate the drawing hierarchy.
- Using *Tools > Expand Design* builds your design based on the views specified in the current expansion configuration. When you expand the drawing, views are derived from the configuration and not from the setup default. After expanding a drawing, you can

navigate the drawing hierarchy using the *File* menu, double-clicking objects, or using the Hierarchy Editor.

The *Next Page* and *Previous Page* icons in the Standard toolbar let you move between the pages of a multipage drawing.

Finding Nets and Cells in Your Design

1. Choose *Tools* > *Expand*.
2. Choose *Tools* > *Global Find*.

   The *Global Find* dialog box appears.
3. In the *Name* box, do one of the following:

   - Type the name of the net or cell to be located. For example, typing ls04 will find all instances of part ls04.

     Note: To find a vectored signal `DATA[3..0]` or `DATA(3..0)`, type `DATA` or `DATA<3..0>` in the *Name* box.

   - Select a previously entered name from the list.

   Select the *Using Wild Card* check box if you want to search using wildcards in the name.
4. Select the object type to be located, either *Net* or *Cell*.
5. To optionally restrict the search by property, perform the following actions in the *With Property* section of the dialog box:

   - Type or select a property name in the *Name* box.
   - Type a property value in the *Value* box or type an * (asterisk) to locate all objects having the specified property name and any property value.

   When you enter the property name and value, they are added to their respective list boxes so that they can be reused during the same design session.
6. Select an option to specify that you want the search results to be listed by full *Hierarchical Names* or by *Library Location*.
7. Click *Find* to begin the search.

   The search process begins. The *Find* button becomes the *Stop Find* button, which you can use to cancel a lengthy search in progress. You can also click *Close*.

A message at the bottom of the dialog box tells you how many instances were found. The (unlabeled) status area box displays the instances of the object found, either by *Hierarchical Names* or *Library Locations*.

8. Choose how you want a selected search result to be viewed: *Zoom to Object*, *Navigate* or both.
9. Click on a search result to view it in your design.

   When you select a result to view, the page containing the object appears with the object highlighted. If you chose *Navigate*, the *Global Navigate* dialog box appears so that you can move across the design to view all net instances listed in the search results box.
10. Perform one of the following steps:

    - Click on another search result to view in the design.
    - Search for a different net or cell by entering a new net or cell name in the *Name* box.
    - Click *Close*.

Navigating Nets in Your Design

1. Choose *Tools > Expand Design*.
2. Choose *Tools > Global Navigate*.

   The *Global Navigate* dialog box appears.
3. Select how the search results are to be viewed, either by full *Hierarchical Names* or by *Library Locations*.
4. Select a net in your design.

   The hierarchical name for the net appears in the *Hierarchical Names* box, and the message box indicates how many nets were located. The status area box lists all located net instances.
5. If you want to zoom in on a search result, select *Zoom to Object*. Otherwise, go to step 6.
6. To view a search result, select a net instance in the status area box.

   When you select a result to view, the page containing the object appears with the object highlighted.

To navigate to a different net, select another net in your design, or click *Close*.

Exiting Concept HDL

➤ Choose *File* > *Exit*.

Concept HDL prompts you whether or not to save any unsaved changes.

Running Commands with Strokes

In the drawing area, place the cursor over an object, or if you are zooming or panning, over the region you want to view.

Note: You can draw stroke patterns, such as undo (U) and zoom fit (W), anywhere in the drawing area, because they do not create a bounding box or act on a specific object.

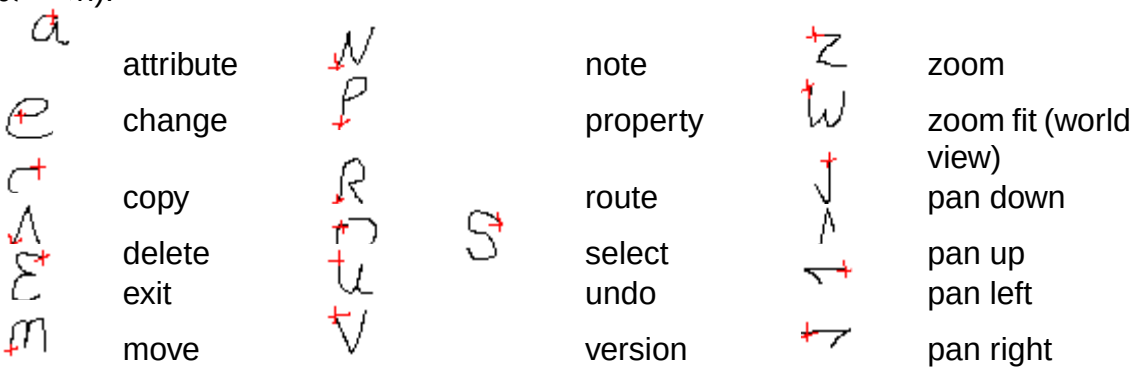
1. Press *Ctrl* or *SHIFT*, hold down the left mouse button and make a stroke.

   You can also press *Ctrl*, hold down the right mouse button and make a stroke.

   Note: If the *Ctrl+LMB Select and Drag* check box in the *General* tab of the *Concept Options* dialog box is selected, you need not press *Ctrl* or *SHIFT*. You only need to hold down the left mouse button and make a stroke.

2. As you move the mouse, you see the stroke pattern being drawn.
3. Release the left mouse button when the stroke is complete.
4. If Concept HDL recognizes the stroke, the command runs as though you typed it into the console window.

Concept HDL provides these default strokes (red indicates the starting point for the stroke pattern):



attribute	note	zoom
change	property	zoom fit (world view)
copy	route	pan down
delete	select	pan up
exit	undo	pan left
move	version	pan right

Guidelines for Strokes

You apply the following guidelines when using strokes:

- Strokes must be entered in the same direction that they were created.
- For strokes that act on a single object, the object under the first point of the stroke is selected.
- For strokes that act on a group of objects, such as zoom (Z) and select (O or S), objects within the first and last points of the stroke are selected.
- Strokes that do not create a bounding box or act on a specific object can be drawn anywhere in the drawing area - for example, zoom fit (W) or undo (U).

Modes in Concept HDL

Depending on the level of design you have opened and the type of editing you want to perform on a design, Concept HDL offers the following three modes:

- In Hierarchy Mode
- Expanded Mode
- Occurrence Edit Mode

In Hierarchy Mode

When you open a design project in Concept HDL, the top-level drawing is displayed and the title bar displays the design name with *in hierarchy* within parentheses. This means that Concept HDL recognizes the design with all its pages and levels. If you have made any changes in the drawing, but have not saved the drawing, Concept HDL shows a * sign in the title bar.

In Hierarchy mode, Concept HDL does not allow you to perform actions such as global find and global navigate. To perform these actions, you must expand the design.

You can ascend or descend into the various pages and levels in a design. You can also use *File > Return* to return to the previous page you had viewed.

When you open any other drawing that is not used in the design and not in the hierarchy of the schematic, Concept HDL opens the drawing with just the design name in the title bar. You cannot use the *File > Return* feature in this case.

Expanded Mode

Concept HDL expands a design to read all pages and levels and to enable communication with other tools. Drawing expansion can take a while to complete, depending on the complexity of design.

To expand your design, choose *Tools* > *Expand Design*. When you have expanded the design, Concept HDL displays the design name with the word *expanded* written within parentheses. Now if you change the drawing on the root design, Concept HDL displays *needs expansion* within parentheses followed by a * sign in the title bar of the root design. If you change the drawing on any other page of the expanded design, Concept HDL displays *needs expansion* on the title bar of the root design.

Once you have expanded your design, you can:

- Highlight (cross-probe) between tools using *Display* > *Highlight* and *Display* > *Dehighlight*.
- View properties from net synonyms that are in the current drawing or in other drawings.
- Choose *Text* > *Attributes* to view net synonyms in expanded drawings.
- Navigate the drawing hierarchy based on the current expansion.
- Find all synonyms of a specified net or all instances of a specified cell in a hierarchical design or multipage schematic. Synonyms are used to specify that two signals with different names are actually the same. You can use the `SYNONYM` symbol in the Standard library to define a synonym.

Occurrence Edit Mode

While netlisting a design, Concept HDL stores the design data in two types of files:

- Connectivity (`verilog.v` or `vhdl.vhd`)
- Properties (`viewprps.prp`)

In a scenario where there are multiple instances of a drawing in a design and you want to specify different properties to each instance in the design, you can use occurrence properties. Occurrence properties can be specified for drawings, components, nets, and pins. The Occurrence Property File (`props.opf`) stores the properties of multiple instances. To edit this file, you have to be in the Occurrence Edit mode. Choose *Tools* > *Occurrence Edit* to shift to occurrence edit mode. You can use the *Attributes* dialog box (*Text* > *Attributes*) in the Occurrence Edit mode to edit the `props.opf` file.

Issuing a command puts Concept HDL in that particular command mode. To end a command (exit a command mode):

- Right-click in the drawing area and choose *Done* in the pop-up menu.
- Type **;** in the console window.
- Press *Esc*.

You can copy and paste text in the console window so that you do not have to recopy commands.

7

Creating a Schematic

You have to perform the following tasks to create a schematic in Concept HDL:

- Create a project.
- Start Concept HDL.
- Create a design page.
- Add a page border.
- Add parts using component browser.
- Connect parts.
- Name signals.
- Add properties.
- Add ports.
- Save the design.
- Work with designs.

Creating a Project

You use Project Manager to create and set up a project. Project Manager creates a project file (<*name of project*>.cpm) that stores paths to local libraries (also known as design libraries), the top-level design name (also known as the root design), part tables (files that map logical components to corresponding physical components), tool settings (defaults), global settings, view directory names, and other related settings for a design project.

A design project consists of the following:

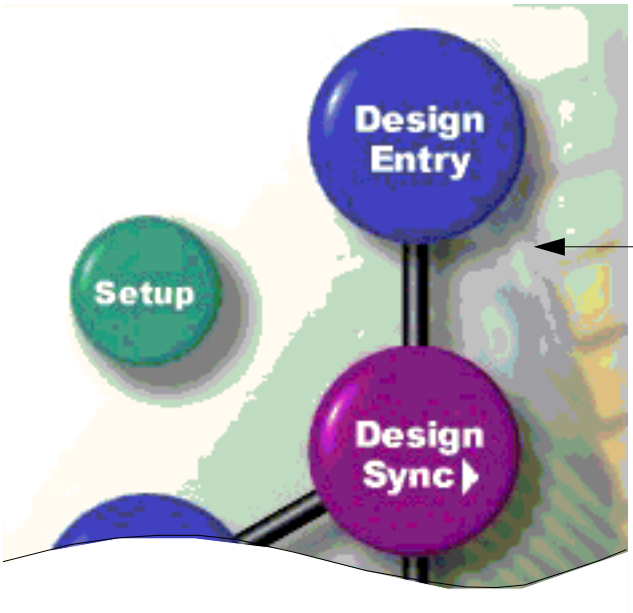
- Reference libraries
- Local libraries

- `cds.lib` file
- Project file (`.cpm` file)

For more information on creating a project using Project manager see “Creating a Project” on page 68.

Starting Concept HDL

After you create the design project in Project Manager, the flow area of Project Manager displays the Cadence Board Design flow. In the Board Design flow, click the *Design Entry* icon.



Click the Design Entry icon to invoke Concept HDL.

About Drawing Names

The drawing name identifies your design. Drawing names can have several fields separated by periods:

```
[<library>]  cell [.view] [.version] [.page]
```

The cell name is the only required part of the drawing name.

Library Name	Identifies the library containing the component. Angle brackets surround a library name.

Cell Name	The cell name describes the intended function of the drawing.
View Name	Describes the representation of a schematic drawing. For a symbol, the view name must be SYM. Standard views in Concept are: Schematic drawings -These can be any name (SCH is the default) and contain symbols that represent library components or hierarchical cells. Symbol drawings-These must be defined as SYM. This is a graphical representation of a library component. It defines the shape, pins, and general properties of the component. When making a hierarchical design, you make a symbol (SYM) drawing to represent an entire schematic (SCH) drawing.
Version	Numbers the schematic or symbol view representation. If you don't specify a version number, Concept HDL assumes version 1 **Note:** Do not use the version field of the drawing name to store revisions of a drawing.
Page	Used for schematics only, the page field identifies the pages of a multi-page drawing.

General Rules to Ensure Compatibility Between Schematics and Other Design Tools

- You should not change the drawing view type when you save the drawing. For example, if you are editing `shifter.sch`, you cannot save it as `shifter.sym`. You must use the *File > Save A*s menu command to change the name and type of a drawing and then save it.
- If you add a component into a symbol drawing, either for comparison purposes or as part of a new symbol, Concept HDL will not let you save the drawing. You must first smash the copied component or group into individual wires, arcs, and notes.

Creating a Design Page

To create a single-page

1. Choose *File > New*.
2. Add a border around the drawing.
3. Choose *File > Save As* to save and name the new drawing.

Use the other system design tools to compile, simulate, and package the design.

To create a multiple-page

1. Choose *File > New*.
2. Place a border in the main design area.
3. Choose *File > Save As* to save and name the new drawing, and specify page 2 in the *Page* field of the *View Save As* dialog box.

Note: Alternatively, while viewing page 1, you can choose *File > Edit Page > Next*, or select the *Next Page* icon in the Standard toolbar.

Use the other system design tools to compile, simulate, and package the design.

Adding Page Borders

Page borders provide a convenient way for documenting information such as the date of creation, the name of the engineer, the page number, the name of the design, and company logo on the schematic. Page borders also serve as a border to demarcate the borders for a schematic.

The Standard library supplied by Cadence contains the following symbols that can be used as page borders:

Page Name	Specifications
A Size Page	8.5 x 11 inch border
B Size Page	11 x 17 inch border
Cadence A Size Page	8.5 x 11 inch border with the Cadence Design Systems logo
Cadence B Size Page	11 x 17 inch border with the Cadence Design Systems logo
C Size Page	17 x 22 inch border
D Size Page	22 x 34 inch border
E Size Page	34 x 44 inch border
F Size Page	44 x 68 inch border

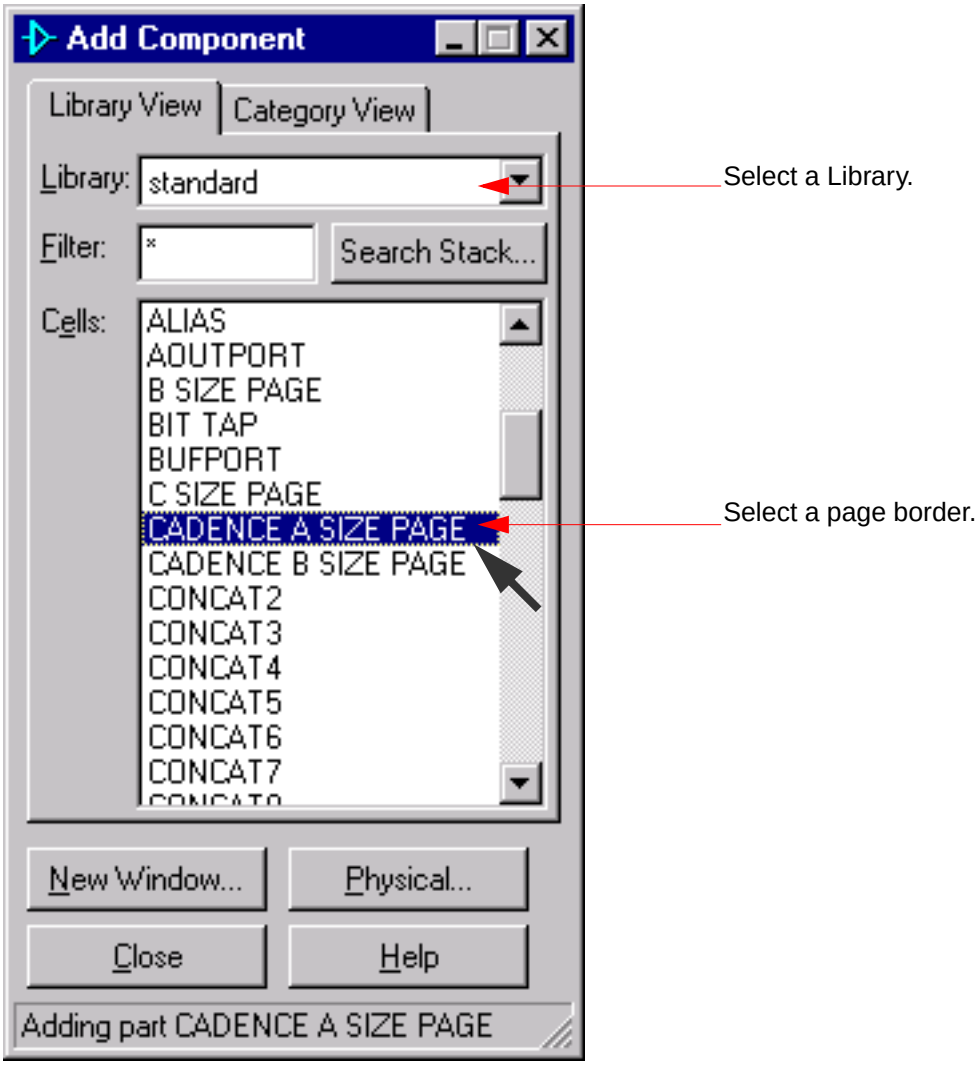
Placing a Border in the Design Area

Concept HDL helps you to automatically add a page border when you create a new page. For more information, see Setting Automatic Page Borders.

To add a page border

1. Choose *Component* > *Add*.

   The *Add Component* dialog box appears.



2. Click the *Library View* tab.
3. In the *Library* box, select the *standard* library.
4. In the scroll area, select a border for your drawing.
5. Click *OK* in the message box that appears.

   The border is attached to the cursor.
6. Place the border in the main design area.

7. Add note text as appropriate in boxes in the lower right corner of the border.

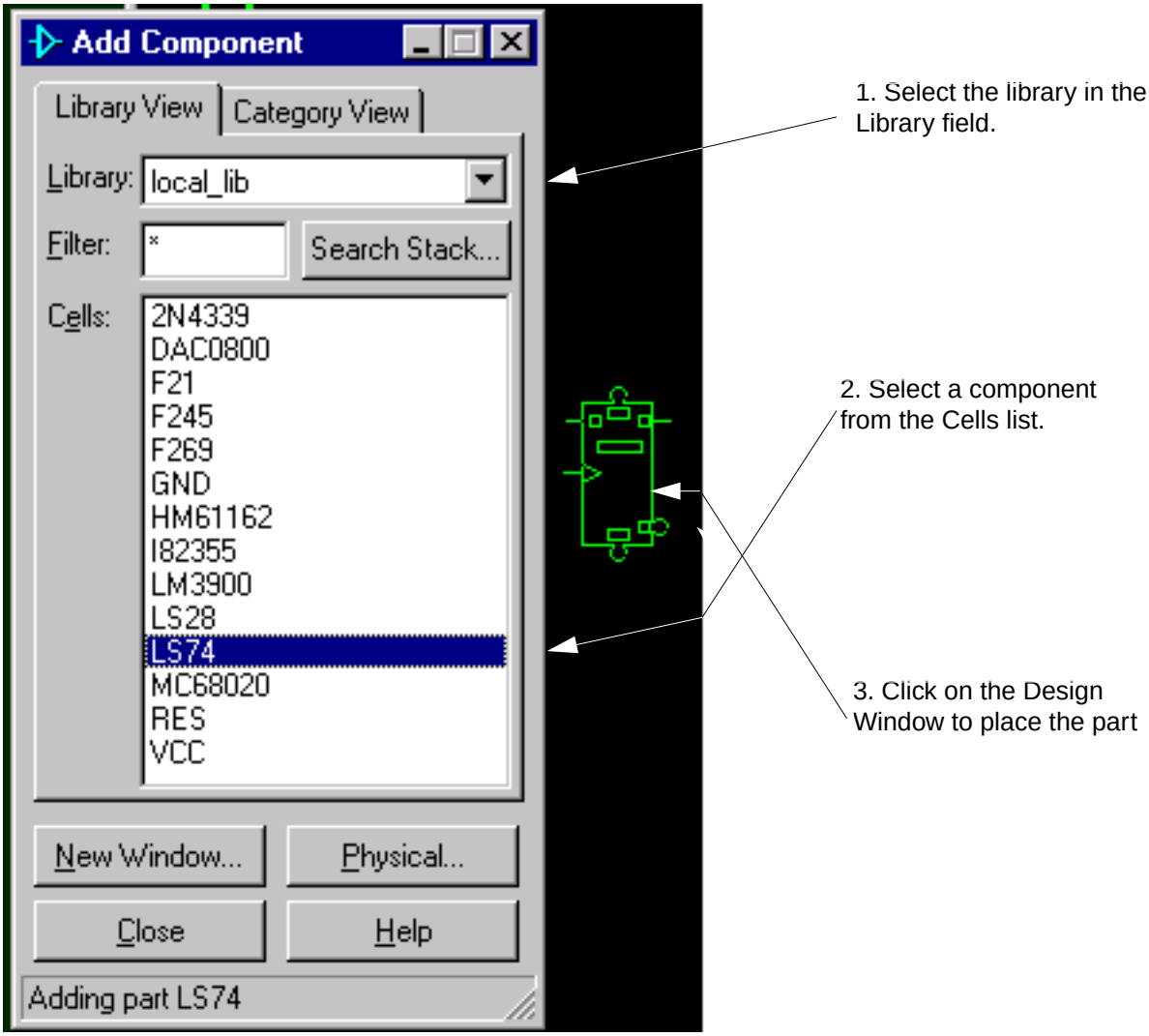
Adding Parts Using the Component Browser

To add parts in Concept HDL, you can use the Component Browser.

To add a part:

1. Choose *Component* > *Add*.

   The *Add Component* dialog box (Component Browser) appears. If you select the *Category* tab, you can select the components under each category.



The *Library View* tab is displayed by default when you open the *Add Component* dialog box. To display the *Category View* tab by default, select the *Show Category View (Add)* check box in the *General* tab of the *Concept Options* dialog box (*Tools > Options > General*).

2. Select a library.

   Concept HDL displays the components in the library you select.

3. Select a component.

   The component attaches to the cursor.

4. Click on the drawing to place the component.

You can continue placing components until you choose another menu item or select *Done* from the pop-up menu.

To add a component with physical information:

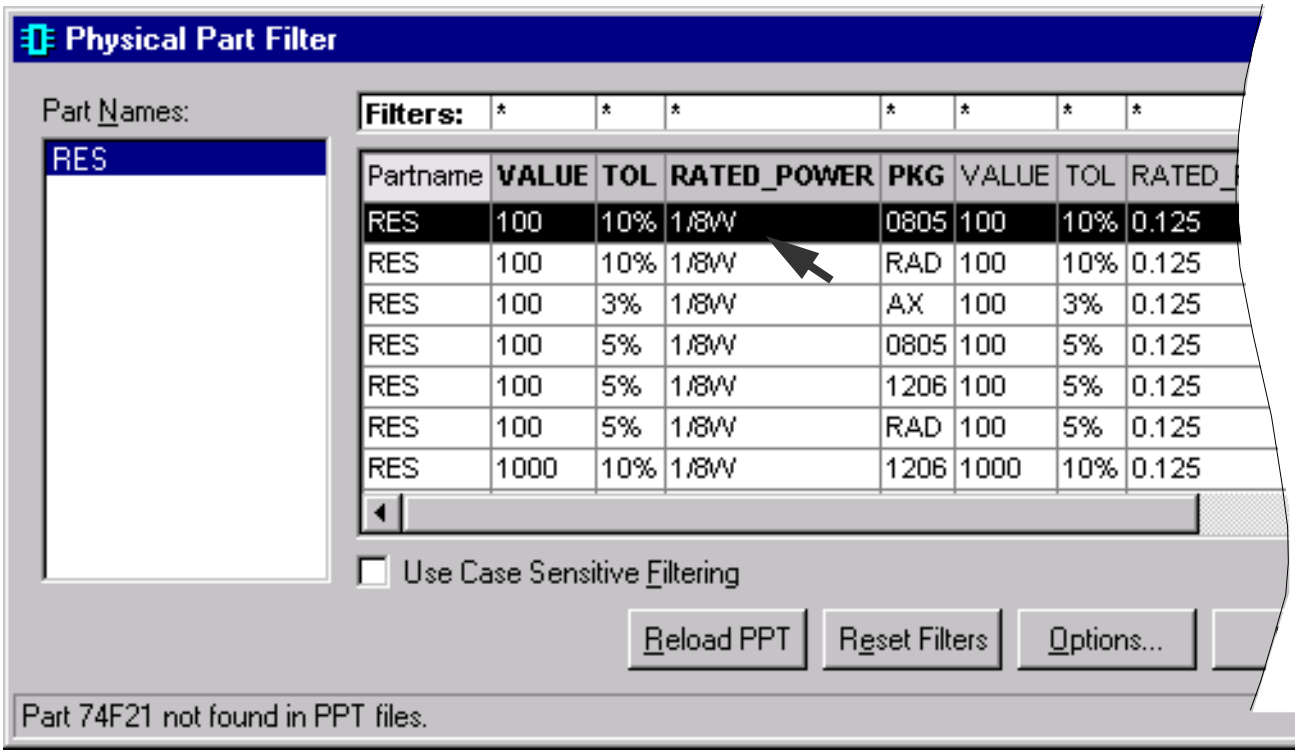
1. Choose *Component > Add* to select a component using the library view or the category view.

2. Click on *Physical*.

   The *Physical Part Filter* dialog box appears.

   Note: To automatically open the *Physical Part Filter* dialog box when you choose *Component > Add*, select the *Show PPT Browser* check box in the *General* tab of

the *Concept Options* dialog box (*Tools* > *Options* > *General*).


Physical Part Filter
Part Names:
RES
Filters:
Partname VALUE TOL RATED_POWER PKG VALUE TOL RATED_
RES 100 10% 1/8W 0805 100 10% 0.125
RES 100 10% 1/8W RAD 100 10% 0.125
RES 100 3% 1/8W AX 100 3% 0.125
RES 100 5% 1/8W 0805 100 5% 0.125
RES 100 5% 1/8W 1206 100 5% 0.125
RES 100 5% 1/8W RAD 100 5% 0.125
RES 1000 10% 1/8W 1206 1000 10% 0.125
Use Case Sensitive Filtering
Reload PPT
Reset Filters
Options...
Part 74F21 not found in PPT files.

The *Physical Part Filter* dialog box displays information from the Physical Part Table file (`.ptf` file). Each row in the *Physical Part Filter* dialog box corresponds to a physical component associated with the logical part you have selected.

If no physical information is available for the part, Concept HDL displays the *Add Part* dialog box, which lets you select the pack type from the `chips.prt` file. Concept HDL annotates the PACK_TYPE property on the schematic.

3. Select a part under *Part Names* in the *Physical Part Filter* dialog box.
4. In the *Filters* row, you can narrow the available part choices based on specific criteria.

   For example, if you want to view only those physical parts of the package type `dip`, type `dip`, or `d*` in the filter space above the pack_type column and press *Enter*.

Note: You can select multiple logical parts in the logical view of the Component Browser and view their physical parts. The multiple selection of logical components and viewing their physical information is most useful when you have selected logical components by category.

To select multiple logical parts randomly, press the Ctrl key and select the parts.

➤ Select a row in the *Physical Part Filter* dialog box and click on the drawing to place the part with physical information.

Concept HDL annotates the schematic with the physical information depending on the options you have set in the *Property Options* dialog box.

Connecting Parts

You can connect parts in Concept HDL using wires. Parts can be connected manually using *Wire > Draw* or automatically connected using *Wire > Route*.

Drawing a Wire Manually

To draw a wire without naming it

1. Choose *Wire > Draw*.
2. Click a pin on a component.
3. Click again wherever you want the wire to bend, or click a pin on another component.

To name a wire when you draw it

1. Choose *Wire > Draw*.
2. Right-click and choose *Signal Name…* from the pop-up menu.
3. Type a signal name in the *Signal Name* box.
4. Click *OK*.
5. Click wherever you want the wire to bend, or click a pin on another component.

Auto-Routing a Wire

1. Choose *Wire > Route*.
2. Click the edge of a component, then click the edge of another component.

Note: You can also run the `route` command using this stroke pattern:

For more information on working with wires, see Chapter 9, “Working with Wires.”.

Naming Signals

To name an existing wire:

1. Choose *Wire > Signal Name*.

   The *Signal Name* dialog box appears.

2. Type one or more signal names on separate lines.
3. Select the wire(s) you are naming in the same order you entered them in the *Signal Name* dialog box.

To name a wire when you draw it:

1. Choose *Wire > Draw*.
2. Right-click and choose *Signal Name…* from the pop-up menu.
3. Type a signal name in the *Signal Name* box.
4. Click wherever you want the wire to bend, or click a pin on another component.

Signal Naming Conventions

The following characters have special significance in signal names. Follow these conventions while naming signals.

Note: These conventions also apply to component path names.

:	Used for concatenating signals. Needs an operand on both sides. For example, A:B is legal, while AB: is illegal.
&	When the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box is selected, an ampersand represents concatenation and needs an operand on both sides. When the *Multi-format Vectors* option is not selected, the ampersand character has no special meaning and can be used anywhere in a signal name.
,	When the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box is selected, a comma represents concatenation and needs an operand on both sides. When the *Multi-format Vectors* option is not selected, a comma has no special meaning and can be used anywhere in a signal name.

\ Must be followed by one of the following characters:

BASE - For declaring the name of the signal as the “base” signal name for all its aliases or synonyms. For more information, see Declaring a Base Signal on page 138.

G - For *Global*. Implies the signal is a global signal. For more information, see Global Signals on page 140.

I - For *Interface*. Implies the signal is a port. Cadence recommends that you use the port symbols INPORT, OUTPORT, or INOUT instead of the \I suffix. The \I suffix declares ports as INOUT ports. If you use the port symbols, you can explicitly declare a port as an IN, OUT or INOUT port. For more information on using port symbols, see Adding Ports on page 132.

L - For *Local*.

/ When used at the beginning of a name, this indicates a global signal.

! When used at the beginning of a name, this indicates a global signal.

{ } Are not special characters and can be used without any restriction.

< > Indicates that the signal is a bus. The angle brackets must be matched correctly and must contain either a parameter or an integer. Cannot be used anywhere else in a signal name.

( ) When the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box is selected, the parentheses indicate that the signal is a bus. Must be matched correctly and contain either a parameter or an integer. When the *Multi-format Vectors* option is not selected, the parentheses have no special meaning and can be used anywhere in the signal name.

[ ] When the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box is selected, the square brackets indicate that the signal is a bus. Must be matched correctly and contain either a parameter or an integer. When the *Multi-format Vectors* option is not selected, the square brackets have no special meaning and can be used anywhere in the signal name.

* When used at the end of a signal name or a pin name, an asterisk indicates that the signal or pin is low-asserted. Cadence recommends that you use the _N suffix to indicate a low-asserted signal or pin. For more information, see Specifying the Assertion Level of Pins and Signals on page 136.

_N When used at the end of a signal name or a pin name, indicates that the signal or pin is low-asserted. For more information, see Specifying the Assertion Level of Pins and Signals on page 136.

~ Cannot be used in signal names.

The following characters can be used in signal names without any restrictions:

.
-
#
$
%
+
=
|
?

Step Size in Signal Names

Bit subscripts specify the number of bits that a signal represents, and identify the bits.

Syntax

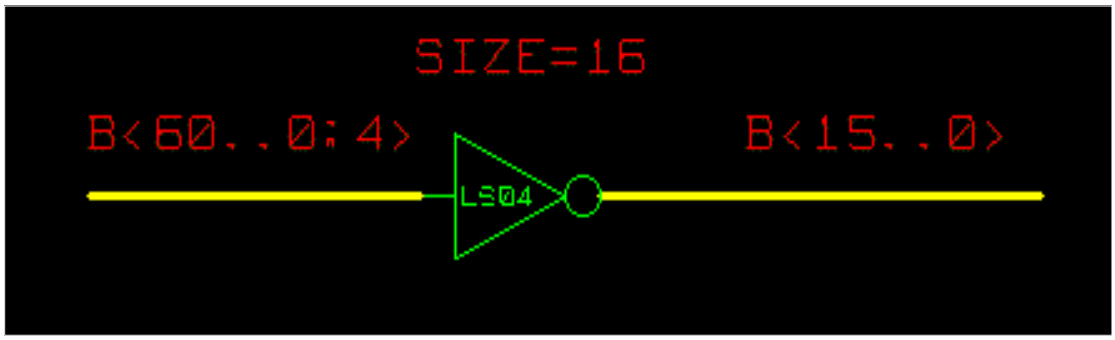
```
<bit1..bit2:step>
```

The syntax specifies a sub-range of bits beginning with bit2 and including every bit that is step bits apart up to bit1. The step value is usually a positive integer. Use a negative integer to reverse the bit order. A step value of 1 is equivalent to no step value.

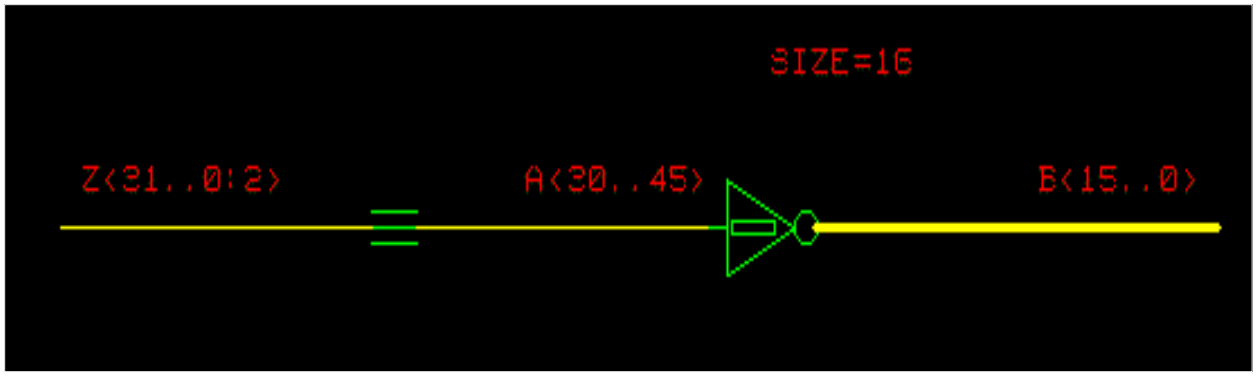
Examples

Subscript	Result
<31..0:2>	30 28 26 ... 6 4 2 0
<11..0:4>	8 4 0
<9..1:3>	7 4 1
<0..31:-1>	31 30 29 ... 3 2 1 0
<15..0:20>	0
<6..0:-2>	6 4 2 0
<7..0:-2>	6 4 2 0

Consider the following examples:



In this example, B<60..0:4> results in
B[60],B[56],B[52],B[48]…B[16],B[12],B[8],B[4],B[0].



In this example, Z<31..0:2> is synonym to A<30..45>.

Z<31..0:2> results in Z[30],Z[28],Z[26],Z[24],..Z[6],Z[4],Z[2],Z[0].
Therefore, Concept HDL does the following assignment:

```
assign a [30:45] =
{z[30],z[28],z[26],z[24],z[22],z[20],z[18],z[16],z[14],z[12],z[10],
z[8],z[6],z[4],z[2],z[0]};
```

Limitations of Signal Naming

- The Concept SCALD syntax <bit:width> is not supported. The syntax <bit:width> is supported as <bit..width>. For example, <31:8> is treated as <31..8>.
- The syntax <bit1..bit2:step> is supported only for signals. This syntax is not supported for pin names. If you create a symbol from a schematic, interface signals become pins and therefore an interface signal cannot use the above syntax. For example, if you have used AA<7..0:2> as the interface signal and created a symbol

from this, Concept HDL will not generate AA<7..0:2> as the pin. It will generate AA<0..7> as the pin.

- The syntax <bit1..bit2:step> is not supported for PATH properties.
- You cannot perform Global Find or Global Navigate operations on buses that have step size in their names. For example, a signal with the name A<31..0:2> is not found by Global Find.
- Bits tapped from a bus that has step size in signal names are not supported. For example, consider a bus named ADDR<11..0:4>. Tapping bits ADDR<8>, ADDR<4>, and ADDR<0> from the ADDR bus ADDR<4..0> is not supported.

Adding Properties

To add one property at a time

1. Choose *Text > Property*.
2. In the *Property* dialog box, enter a name in the *Property Name* box and a value in the *Property Value* box.
3. Click *OK*.
4. Click the object to which you are attaching the property.
5. Click near the object to indicate where to display the property information.

   As the default, Concept HDL displays only the property value. Choose *Text > Property Display* to modify how properties are displayed.

Note: You can also run the `property` command using this stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes on page 112.

You can also add a property using the property command.

To add many properties at the same time using the Attributes dialog box

1. Choose *Text > Attributes* or type `attribute` in the console window.

2. Select an object.

   The object you select appears highlighted, and the *Attributes* dialog box displays attributes for the object.

3. Click *Add* in the *Attributes* dialog box.

   An empty row appears.

4. Type a property name and a value in the appropriate columns.

5. Adjust property visibility and alignment as needed.

6. Click *OK* to add, or click *Cancel*.

To add occurrence properties

1. Choose *Tools > Expand Design*.

2. Choose *Tools > Occurrence Edit* or type `attribute` in the console window.

3. Choose *Text > Attributes*.

4. Select an object.

   The object you select is highlighted, and the *Attributes* dialog box displays attributes for the object.

5. In the *Attributes* dialog box, click *Add*.

   An empty row appears.

6. Enter a property name in the *Name* box and a property value in the *Occurrence Value* box.

   You can also replace the occurrence value for an existing property. However, you cannot change or delete the schematic values that are shaded gray.

7. Adjust property visibility and alignment as needed.

8. Click *OK*.

9. Click near the object to indicate where to display the property information.

   By default, Concept HDL displays only the property value. Choose *Text > Property Display* to modify how properties are displayed.

Adding Ports

When you are creating a Concept HDL schematic, you must place port symbols on the page to indicate the ports on the entity. Although a signal name with a `\I` suffix is acceptable, it is preferable to use a port symbol instead. The Standard library has the following port symbols:

- INPORT (input)
- IOPORT (bi-directional: Input/Output)
- BUFPORT (used only for VHDL)
- OUTPORT (output)
- LNKPORT (used only for VHDL)
- AOUTPORT (for behavioral assignments)

Using PORT Symbols

To use a port symbol:

1. In Concept HDL, choose *Component* > *Add*.

   The *Component Browser* appears.
2. Select `standard` from the *Library* drop-down list.
3. Select a port symbol from the *Cells* list.
4. Click in the Concept HDL drawing area to place the symbol.
5. Close the *Component Browser*.
6. Attach a wire to the pin on the port and connect it to an instance.

   Note: A VHDL and Verilog restriction prohibits you from wiring different ports of an entity together. Concept HDL gives a warning if different ports of an entity are wired together in your schematic.
7. Choose *Wire* > *Signal Name* and name the wire.

   This signal name is the port declaration in the VHDL and Verilog text.
8. Define the VHDL logic type and Verilog logic type of the port.

   The default VHDL logic type for ports in Concept HDL schematics is `STD_LOGIC` for scalar ports and `STD_LOGIC_VECTOR` for vectored ports. The default Verilog logic type for all ports in Concept HDL schematics is `WIRE`. You can change these defaults for a

drawing or for the entire project. You can also override these default logic types by choosing a different type for individual ports. See Setting the Verilog Logic Type for Ports and Signals on page 157 and Setting the VHDL Logic Type for Ports and Signals on page 160 for details.

If you plan to use custom port symbols instead of those supplied in the Standard library, copy all the visible and invisible properties from the port symbols to the new symbol.

Note the following when you use port symbols:

- If you leave an output port of an instance unconnected or if you attach the signal `NC` to a port, the port is represented as open in VHDL and as unconnected in Verilog.
- Although the VHDL language allows you to create ports with an unconstrained range, you cannot create Concept HDL schematics with unconstrained ports. If you want an unconstrained port, use a parameterized range for the port. For more information, see Unconstrained Ranges for Ports, Signals, and Aliases on page 165.
- To connect a signal of one type to a port of another type in VHDL, use a type conversion function. Verilog does not use type conversion functions; in Verilog you can connect a signal of type `WIRE` to ports of other types. While generating Verilog text, Concept HDL ignores any type conversion properties that you place on the schematic for VHDL. For more information about type conversion, see Type Conversion on page 166.
- Verilog does not support abstract data types. In VHDL, however, ports can be of abstract data types such as integers and floating point numbers. For more information, see Abstract Data Types in VHDL on page 167.

Rules for Using Port Symbols

- Name the signal to which a port is attached.
- Do not leave an input port of an instance unconnected; it will generate both a VHDL error and a Verilog error.
- Do not connect ALIAS symbols to ports.
- Do not wire different ports of an entity together.
- Follow port association rules. For more information, see Port Association Restrictions.

Port Association Restrictions

The VHDL language has strict rules regarding the port associations allowed between ports of component instances within an architecture and the ports of the entity declaration.

Port association rules are not as strict for Verilog as they are for VHDL. If you use the VHDL_USER=NO property on the VERILOG_DECS symbol, you do not need to follow the rules described here.

The following table shows the port associations allowed in VHDL.

Formal Port	Actual Port
IN	IN, INOUT, BUFFER
OUT	OUT, INOUT
INOUT	INOUT
BUFFER	BUFFER

A formal port is the port on an instance; an actual port is the port in the entity description.

For example, if a formal port is an INOUT port and it is connected to ports higher up in the design hierarchy, the other ports must also be declared as INOUT ports. Similarly, BUFFER ports must remain BUFFER ports as they ascend the design hierarchy.

Working with Ports and Signals

This section describes the following:

- Setting the Initial Value of a Signal on page 135
- Specifying the Assertion Level of Pins and Signals on page 136
- Creating an Alias for a Signal on page 137
- Global Signals on page 140
- Unnamed Signals on page 147
- Signal Concatenations on page 147
- Signal Replication on page 150
- Merge Symbols on page 150
- Signal Slices (Bit and Part Selects) on page 154
- Setting the Verilog Logic Type for Ports and Signals on page 157
- Setting the VHDL Logic Type for Ports and Signals on page 160

- Specifying Ranges for Ports, Signals and Aliases on page 163
- Unconstrained Ranges for Ports, Signals, and Aliases on page 165
- Resolved Types and Resolution Functions on page 165
- Type Conversion on page 166
- Abstract Data Types in VHDL on page 167

Setting the Initial Value of a Signal

To specify the initial value of a signal for VHDL, you must attach the VHDL_INIT property to the signal. To do this:

1. In Concept HDL, choose *Text* > *Attributes*.
2. Click on the signal to display the *Attributes* dialog box.
3. Click *Add*.
4. Type VHDL_INIT in the *Name* text box and the initial value of the signal in the *Value* text box.

   Note: Do not use quotes around the value. Concept HDL automatically adds quotes to the value.
5. Click *OK* to save the changes and close the *Attributes* dialog box.

Note: The VHDL_INIT property can also be attached to the pins of a symbol. When the VHDL_INIT property is attached to a power symbol or to its pin, the power signal is assigned that value in the VHDL text created by Concept HDL.

Specifying the Size of Nets

Use the SLASH symbol in the Standard library to specify the size of nets. The SLASH symbol is useful for:

- Documenting the width of signals. Normally, the width of a signal is apparent from the bit subscript on the signal name. However, if the signal name is not visible on a certain part of the schematic, you can use the SLASH symbol to document the width. The width you specify on the SLASH symbol must be the same as the actual width of the signal.
- Specifying the size of nets that do not have a size. For example, if you want to make an unnamed net a bus, you can put a SLASH symbol on the net and specify its size. Similarly,

you can use a SLASH symbol to specify the width of unnamed outputs of CONCAT and MERGE symbols.

Using a SLASH Symbol

To use a SLASH symbol:

1. In Concept HDL, choose *Component* > *Add*.

   The *Component Browser* appears.
2. Select standard from the *Library* drop-down list.
3. Select the SLASH symbol from the *Cells* list.
4. Click in the Concept HDL drawing area to place the symbol.
5. Close the *Component Browser*.
6. Attach the SLASH symbol to the net you want to size.
7. From the *Text* menu, choose *Attributes*.
8. Click on the SLASH symbol to display the *Attributes* dialog box.
9. Specify the size of the net by changing the value of the SIZE property. The default value of the SIZE property is 1.

   Note: If you are using a SLASH symbol only for documentation and the width of the signal is already set, the value of the SIZE property on the SLASH symbol must match the actual width of the signal.
10. Click *OK* to save the changes and close the *Attributes* dialog box.

Specifying the Assertion Level of Pins and Signals

Cadence recommends a notation for representing the assertion level of pins and signals. By convention, a signal is active high for positive logic and active low for negative logic. Two signals with the same name but with different assertion levels are considered to be different signals.

The assertion level of a signal is determined by the _N or * suffix or a - prefix. Cadence recommends that you use a _N suffix to indicate a low-asserted signal or pin.

To specify a low-asserted signal or pin

➤ Suffix _N or * to the signal name or pin name.

Example

ENABLE_N is an active low scalar signal.

DATA<15..0>_N is an active low vector signal.

ENABLE* is an active low scalar signal.

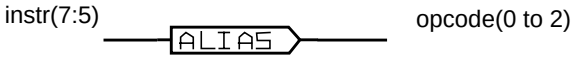
DATA<15..0>* is an active low vector signal.

Any signal or pin that does not have the _N or * suffix is assumed to be an active high signal.

Creating an Alias for a Signal

Use the ALIAS symbol in the Standard library to specify another name for a signal.

For example, the following ALIAS



creates the following alias declaration in VHDL

```
alias opcode: std_logic_vector (0 to 2) is instr (7 downto 5);
```

and the following alias declaration in Verilog

```
alias_vector alias_inst1 (opcode[0:2], instr[7:5]);
defparam alias_inst1.size = 3;
```

The ALIAS symbol is similar to the SYNONYM symbol in the Standard library. If you use SYNONYM symbols, Concept HDL will not detect all the VHDL- related errors and your VHDL output will be inaccurate. Therefore, if you want to generate VHDL text from the schematic, you must use ALIAS symbols instead of SYNONYM symbols.

If you currently use SYNONYM symbols in your designs, replace them with ALIAS symbols. If you are not generating VHDL text from your schematic, you can use either the ALIAS symbol or the SYNONYM symbol.

Creating an ALIAS

To create an ALIAS

1. In Concept HDL, choose *Component* > *Add*.

   The *Component Browser* appears.
2. Select `standard` from the *Library* drop-down list.
3. Select `ALIAS` from the *Cells* list.
4. Click in the Concept HDL drawing area to place the symbol.

5. Close the *Component Browser*.
6. Attach the original signal to the left pin of the ALIAS symbol.
7. Attach the ALIAS signal name to the right pin of the ALIAS symbol.

Rules for Using ALIAS Symbols

ALIAS symbols

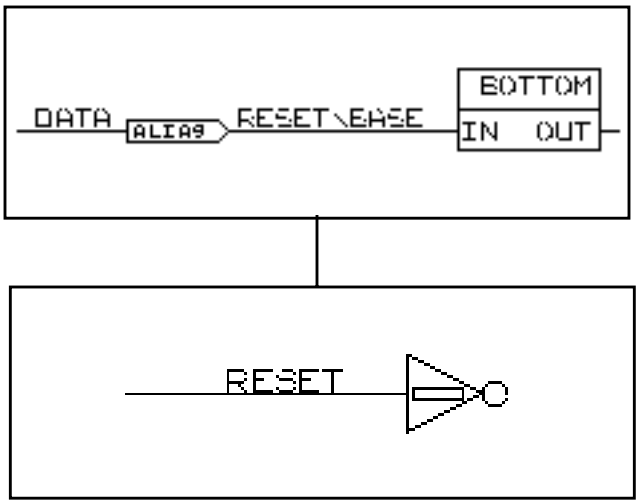
- can be connected to a SLICE.
- should not be connected to ports.
- should not be connected to global signals.
- should not be connected to the output of taps.
- should not be connected to the output of CONCAT signals.
- should not be connected to unnamed signals.

Declaring a Base Signal

When two signals are aliased or synonymed, Concept HDL selects one of the signal names as the base signal. The name of the base signal becomes the name of corresponding physical net in Allegro. A signal is its own base signal if it is not aliased or synonymed to any other signal or if it is selected as the base signal.

You may want the name of a particular aliased or synonymed signal to be passed to Allegro as the physical net name. You can force Concept HDL to use a particular aliased or synonymed signal by declaring it as a base signal, as below:

➤ Suffix \BASE to the signal name or add the MAKE_BASE=TRUE property on the signal at the highest schematic level at which the signal exists.



Schematic for block MID

Schematic for block BOTTOM

For example, in the above hierarchical design, signal DATA is aliased to signal RESET in the schematic for block MID. The RESET signal is also present in the schematic for the lower level block BOTTOM. To declare signal RESET as the base signal, you must suffix \BASE or add the MAKE_BASE=TRUE property on the signal RESET in the schematic for block MID and not on the signal RESET in the schematic for the lower level block BOTTOM.

Because aliased or synonymed signals at higher schematic levels always supercede aliased or synonymed signals at lower schematic levels, it is only meaningful to use the \BASE suffix or add the MAKE_BASE=TRUE property to the aliased or synonymed signal at the highest schematic level at which the signal exists.

Note: If a global signal is aliased or synonymed to a signal that has the \BASE suffix or the MAKE_BASE=TRUE property, the global signal will always be treated as the base signal. For more information, see Rules for Choosing the Base Signal on page 140.

Note: If a power symbol vcc is aliased or synonymed to another power symbol or global signal 5V and you want to make the 5V signal as the base signal, name the signal that is connected to the power symbol 5V as 5V\BASE\G, where \G represents a global signal. For more information on global signals, see Global Signals on page 140.

Rules for Choosing the Base Signal

The rules for selecting a base signal are listed below, in the order in which Concept HDL applies them. If Concept HDL cannot select the base signal name from the first rule, the next rule is applied, and so on.

1. Select a global signal over a non-global signal.

   For example, if a signal `CLOCK` is aliased to a global signal `SWITCH\G`, the global signal will be the base signal.

   Note: The global signal will be the base signal even if is aliased or synonymed to a non-global signal that has the `\BASE` suffix or the `MAKE_BASE=TRUE` property.

2. Select the signal that has the `\BASE` suffix in its name or has the `MAKE_BASE=TRUE` property.

3. Select a constant signal over a non-constant signal.

   For example, if a constant signal `123` is aliased to a non-constant signal `CLOCK`, the constant signal will be the base signal.

4. Select the lower bit number of two signals with the same name (for example, `X<0>` is selected over `X<3>`).

5. Select a user-assigned signal name over an unnamed signal.

   For example, if a signal `CLOCK` is aliased to an unnamed signal, the signal `CLOCK` will be treated as the base signal.

6. Select a scalar signal over a vector signal.

7. Select the signal that is lexicographically smaller (for example, `CLK` is selected over `CLOCK`).

Global Signals

If you want to use a global signal in your schematic, suffix `\G` to the signal name.

Verilog global signals modules and VHDL global packages are created automatically for you from schematics that contain global signals; you do not have to create these manually. Global modules and packages are created when you either expand a design or package it.

When you expand or package, a cell called `glbl` is created in the design library. This cell has `RootDesign_Configuration` views, which contain a `verilog.v` file with the Verilog global signal module and a `vhdl.vhd` file with the VHDL globals package.

Shorting of Global Signals

If a global signal in your design is shorted with another global signal, errors are displayed when you save or package the design. For example, if a +5V global is shorted with a GND global signal:

- The following error message is displayed in the *Markers* dialog box when you save the design:

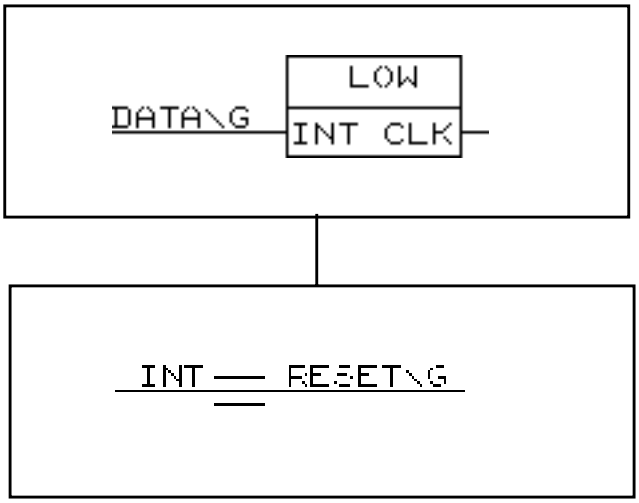
  ```
  ERROR:275: Two global signals are shorted.
  ```

- The following error message can be seen in the Export Physical *Progress* window or in the `pxl.log` file when you package the design:

  ```
  Two global signals are shorted. If you want to short these global
  signals, add them in Allowed Global Shorts list in ConceptHDL setup
  options.
  Net name: '@<root_design_name>.glbl(<root_design_name>_cfg_package):+5v'
  Net name: '@<root_design_name>.glbl(<root_design_name>_cfg_package):GND'
  ```

You might have intentionally shorted some global signals in your design. If you want to allow such global signals to be shorted, add them in the *Allowed Global Shorts* list of the *Concept HDL Options* dialog box. Concept HDL will not display this error message if the global signals listed in the *Allowed Global Shorts* list are shorted.

When you save a design in Concept HDL, error messages are displayed only for the global signals that are shorted within the block you are currently editing. When you package the design, error messages can be seen in the Export Physical *Progress* window or in the `pxl.log` file for the global signals that are shorted across different blocks in the design.



Schematic for block MID

Schematic for block LOW

In the above figure, signal DATA\G is connected to the pin INT of block LOW in the schematic for block MID. In the schematic for block LOW, signal INT is synonymed with signal RESET\G. This results in the shorting of the global signals DATA\G and RESET\G across the blocks in the design. When you save the schematic for block MID or the schematic for block LOW in Concept HDL, no error message for global signal short is displayed. However, when you package the design, error messages for the shorting of global signals DATA\G and RESET\G can be seen in the Export Physical *Progress* window or in the pxl.log file.

Using the Allowed Global Shorts List

For example, suppose that there are 5 global signals—GND\G, +5V\G, DATA\G, CLOCK\G and SWITCH\G—in a design, where:

- Signal +5V\G gets aliased to signal GND\G by mistake
- Signal +5V\G is synonymed to signal DATA\G to intentionally short the signals
- Signal CLOCK\G gets synonymed to signals SWITCH\G and DATA\G, where signals CLOCK\G and SWITCH\G are intentionally shorted, but signal CLOCK\G got synonymed to signal DATA\G by mistake

To allow signal +5V\G to be shorted with signal DATA\G and signal CLOCK\G to be shorted with signal SWITCH\G, do the following:

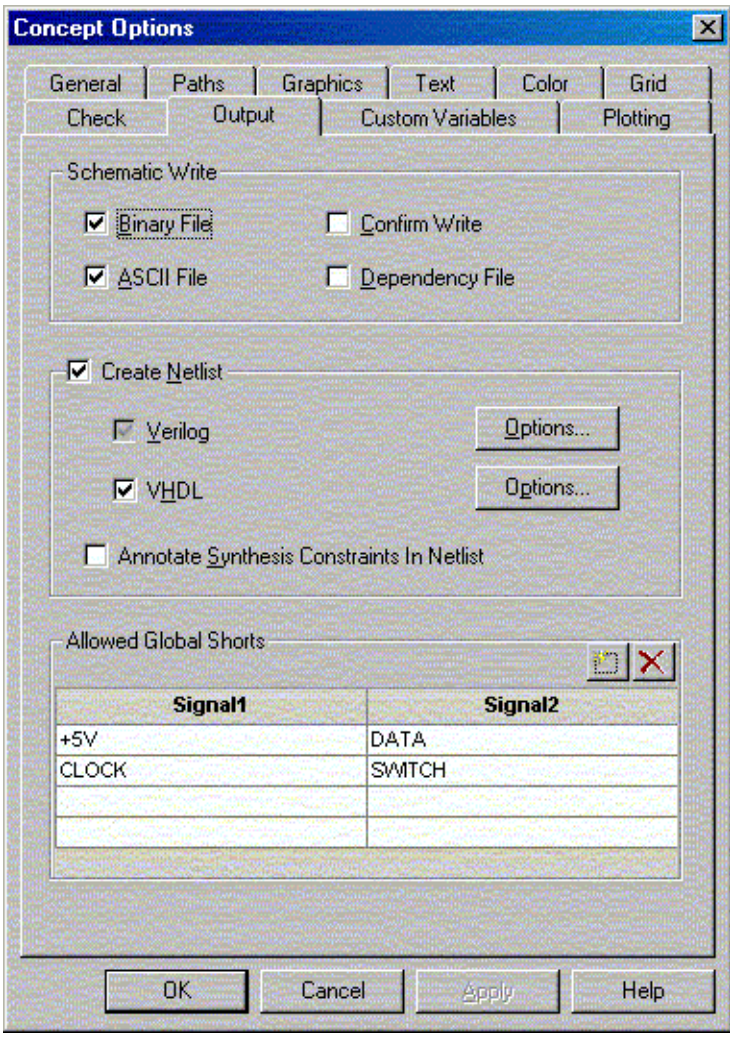
1. Choose *Tools* > *Options* in Concept HDL.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab and do the following in the *Allowed Global Shorts* list:

   a. Type +5V in the *Signal1* field and DATA in the *Signal2* field next to the +5V signal.

b. Type `CLOCK` in the *Signal1* field and `SWITCH` in the *Signal2* field next to the `CLOCK` signal.



The *Allowed Global Shorts* list allows you to add pairs of global signals that you want to remain shorted in the design. Click to add a row. Select a row and click to delete the row.

3. Click *OK*.

When you save or package the design, error messages are displayed only for the shorted global signals that are not specified in the *Allowed Global Shorts* list.

If two shorted global signals are specified in the *Allowed Global Shorts* list and if:

- you have specified one of the signals in the *Supply 0* list in the *Verilog* netlisting options dialog box or have added the `VLOG_NET_TYPE=SUPPLY0` property on it, and

- you have specified the other signal in the *Supply 1* list in the *Verilog* netlisting options dialog box or have added the VLOG_NET_TYPE=SUPPLY1 property on it,

Concept HDL displays the following error message when you netlist the design for digital simulation using *Tools > Simulate*:

```
ERROR:275: Two global signals are shorted.
```

This error is generated because in a design, *Supply 0* and *Supply 1* signals should not be shorted.

For example, if you have specified two shorted global signals CLOCK\G and SWITCH\G in the *Allowed Global Shorts* list and have also added CLOCK\G in the *Supply 0* list and SWITCH\G in the *Supply 1* list in the *Verilog* netlisting options dialog box, the above error message is displayed by Concept HDL when you netlist the design for digital simulation.

Note: This error message will not be displayed when you save or package the design because the shorted global signals are specified in the *Allowed Global Shorts* list.

For more information on setting up Verilog netlisting options and netlisting the design for digital simulation, see Netlisting for Digital Simulation on page 376.

Supported Syntax in the Allowed Global Shorts List

The syntax that is supported in the *Allowed Global Shorts* list is explained below using examples:

Example	Enter the following in Signal1 field of the Allowed Global Shorts list	Enter the following in Signal2 field of the Allowed Global Shorts list
■ To allow shorting of a scalar global signal CLOCK\G with another scalar global signal DATA\G	CLOCK	DATA
■ To allow shorting of any bit of a vectored global signal CLOCK<3..0>\G with any bit of another vectored global signal DATA<4..0>\G	CLOCK	DATA

Example	Enter the following in Signal1 field of the Allowed Global Shorts list	Enter the following in Signal2 field of the Allowed Global Shorts list
■ To allow shorting of two vectored global signals `INT<3..0>\G` and `DATA<4..7>\G` that have the same width This means that Concept HDL allows shorting of bit `INT<3>` with bit `DATA<4>`, `INT<2>` with `DATA<5>`, `INT<1>` with `DATA<6>` and `INT<0>` with `DATA<7>`. **Note:** Concept HDL displays an error message if the width of the shorted signals specified in the *Allowed Global Shorts* list is not the same.	INT<3..0>	DATA<4..7>
■ To allow shorting of the third bit of a vectored global signal `CLOCK<3..0>\G` with the fifth bit of another vectored global signal `DATA<7..0>\G` The shorting of any other bit of `CLOCK` with any other bit of `DATA`, except `CLOCK<3>` and `DATA<5>` is reported as a global signal short error.	CLOCK<3>	DATA<5>
■ To allow shorting of any bit of a vectored global signal `DATA<3..0>\G` with a scalar global signal `VCC\G`	DATA<3..0>	VCC
You can also use the following syntax to allow shorting of any bit of a vectored global signal `CLOCK<3..0>\G` with a scalar global signal `DATA\G`	CLOCK	DATA
■ To allow shorting of a bit of global signal `CLOCK(3)\G` with another bit of a global signal `DATA[5]\G`.	CLOCK<3>	DATA<5>

Example	Enter the following in Signal1 field of the Allowed Global Shorts list	Enter the following in Signal2 field of the Allowed Global Shorts list
Note: If the *Multi-format Vectors* check box in the *General* tab of the *Concept Options* dialog box is not selected, the signals with the above syntax are treated as scalar signals. To short the scalar global signal `CLOCK(3)\G` with the scalar global signal `DATA[5]\G` For more information on the *Multi-format Vectors* option, see the *Concept HDL Online Help*.	CLOCK(3)	DATA[5]

Note the following when adding signals in the *Allowed Global Shorts* list:

- Shorting is allowed only between a pair of signals specified in the *Allowed Global Shorts* list and not between all the signals in the *Allowed Global Shorts* list.

  For example, suppose that signal `DATA\G` is aliased to signal `RESET\G` and signal `RESET\G` is aliased to signal `CONTROL\G`. If you add signals `DATA` and `RESET`, and signals `RESET` and `CONTROL` in the *Allowed Global Shorts* list, it does not inherently mean that the shorting of signals `DATA` and `CONTROL` is allowed. Concept HDL will not display any error message for global signal short when you save the design. However, when you package the design, error messages can be seen in the Export Physical *Progress* window or in the `pxl.log` file that the global signals `DATA\G` and `CONTROL\G` are shorted. To allow shorting of signals `DATA\G` and `CONTROL\G`, type `DATA` in the *Signal1* field and `CONTROL` in the *Signal2* field next to the `DATA` signal.

- Suppose that two vectored global signals `INT<2..0>\G` and `CLOCK<2..0>\G` are shorted. To allow shorting of the signals, you must specify `INT<2..0>` and `CLOCK<2..0>` in the *Allowed Global Shorts* list. If you specify the shorting of bits:

  - `INT<2>` with `CLOCK<2>`
  - `INT<1>` with `CLOCK<1>`
  - `INT<0>` with `CLOCK<0>`

  in the *Allowed Global Shorts* list, the global signal short error will not be supressed. Also, if you specify the shorting as below in the *Allowed Global Shorts* list, the global signal short error will not be supressed:

- ❑ INT<2> with CLOCK<2>
- ❑ INT<1..0> with CLOCK<1..0>

■ Suppose that the *Multi-format Vectors* check box in the *General* tab of the *Concept Options* dialog box is deselected and you have a scalar global signal CLOCK(3)\G aliased or synonymed to another scalar global signal INT[4]\G in your schematic. If you want to allow the signals to be shorted, add the signals in the *Allowed Global Shorts* list using the same signal syntax—CLOCK(3) and INT[4].

If you now select the *Multi-format Vectors* check box, Concept HDL treats the signals CLOCK(3)\G and INT[4]\G as vectored signals. When you save or package the design, Concept HDL displays an error message that the two global signals CLOCK<3> and INT<4> are shorted. To correct this problem, modify the syntax of the signal names to CLOCK<3> and INT<4> in the *Allowed Global Shorts* list (when the *Multi-format Vectors* check box is selected).

For more information on the *Multi-format Vectors* option, see the *Concept HDL Online Help*.

Unnamed Signals

If you have unnamed signals in your schematic, Concept HDL converts them to UNNAMED_n, where n is a unique number.

Note: You can use unnamed nets with the NOT, CONCAT, and MERGE symbols only if you specify the size of the net with a SLASH symbol.

Signal Concatenations

Signal concatenations allow you to merge a group of signals, ports, or signal aliases into a group. You can then route this group of signals to ports with a single wire.

You can create signal concatenations in Concept HDL schematics either textually or graphically (with the CONCAT symbols in the Standard library). For more information, see CONCAT Symbols on page 621.

To concatenate signals, ports, or signal aliases textually

➤ Attach a signal name to a wire and type the name of the two signals separated by a colon (:).

If you select the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box, you can also use a comma (,) or an ampersand (&) to concatenate

two signals. When the *Multi-format Vectors* option is not selected, ampersands and commas do not represent concatenation and have no special meaning in a signal name. For more information, see Naming Signals on page 126.

A concatenated signal of this type must be connected to a pin of the same width.

For example, if you have two signals `hi_addr(15:0)` and `lo_addr(15:0)` you can attach the following signal name to a wire.

```
hi_addr(15:0):lo_addr(15:0)
```

The wire now represents a 32-bit signal.

For vectored signals, VHDL and Verilog have leftmost and rightmost bits. For the 32-bit signal example above, the leftmost bit is `hi_addr(15)` and the rightmost bit is `lo_addr(0)`.

You can also create concatenations that contain more than two signals. For example, the following concatenation creates a 21-bit signal, assuming that `b` is a scalar signal. The leftmost bit is `a(0)`, and the rightmost bit is `c(0)`.

```
a(0 to 9):b:c(9 downto 0)
```

In addition to regular signals, a concatenation can include ports from the entity for this architecture as well as aliases for other signals. The output of a concatenation can be named. You can also feed the result of one concatenation (or slice) into the input of another concatenation.

Note: You can use unnamed nets with CONCAT symbols only if you size the net using the SLASH body.

To concatenate signals, ports, or signal aliases graphically

1. In Concept HDL, choose *Component* > *Add*.

   The *Component Browser* appears.

2. Select `standard` from the *Library* drop-down list.

3. Select a CONCAT symbol from the *Cells* list.

   For more information on CONCAT symbols, see CONCAT Symbols on page 621.

4. Click in the Concept HDL drawing area to place the symbol.

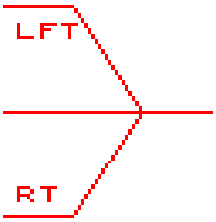
5. Close the *Component Browser*.

6. Wire the signals you want to concatenate to the pins on the left side of the CONCAT symbol. You can also attach signal names directly to the pins.

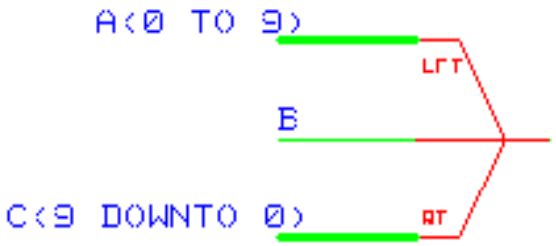
7. Wire the right pin of the CONCAT symbol to the instance to which you want to connect the concatenation.

Example: Concatenating 3 Signals

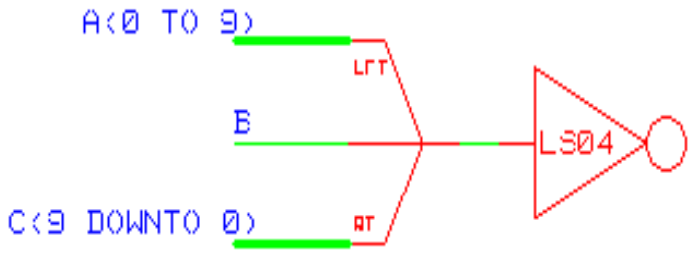
1. Add the CONCAT3 symbol from the Standard library.



2. Wire the signals you want to concatenate to the left pins. In this example, the signals are the vectored signal `a(0 to 9)`, the scalar signal b, and the vectored signal `c(9 downto 0)`.



3. Wire the right pin to the instance to which you want to connect the concatenation.



The output of the following concatenation is the 21-bit signal:

```
a(0 to 9) & b & c(9 downto 0)
```

You can cascade the output of one concatenation into the input of another. The output of the above CONCAT3 symbol can be connected to the input of another CONCAT symbol.

Signal Replication

Concept HDL does not support signal replication notation such as \R. To construct a signal replication, use either textual or graphical concatenations, or a combination of both. For example, if you want a 10-bit wide concatenation of the signal `GND`, do one of the following:

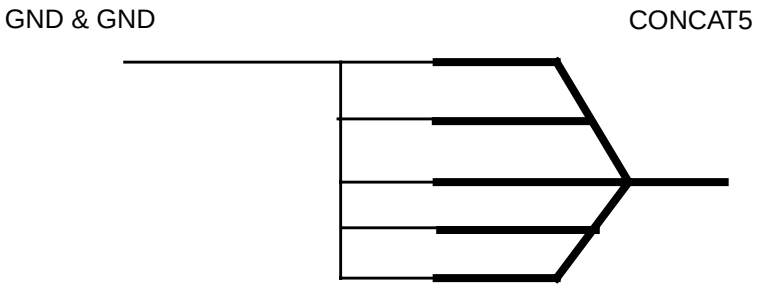
- Name the signal:

  ```
  GND : GND : GND : GND : GND : GND : GND : GND : GND : GND
  ```

- Use the CONCAT10 symbol.

  For more information on CONCAT symbols, see CONCAT Symbols on page 621.

- Use a combination of text and graphics.



The output of one concatenation can also be attached to the input of another concatenation. For example, if the output of the CONCAT5 symbol shown above is attached to the input of a CONCAT4, the resulting width of the signal will be 40 (2 x 5 x 4).

Merge Symbols

Use a MERGE symbol in the Standard library to combine several signals into a single vectored signal, or separate a vectored signal into a number of separate signals. With MERGE symbols, you can draw a vectored signal (a bus) as a single wire in parts of the drawing, and as several signals in other parts of the drawing.

There are nine MERGE symbols in the Standard library: 2 MERGE, 3 MERGE, 4 MERGE, 5 MERGE, 6 MERGE, 7 MERGE, 8 MERGE, 9 MERGE, and 10 MERGE1. Use 2 MERGE to merge two signals, 3 MERGE to merge three signals, and so on. Each MERGE symbol has

four versions: two for merging signals and two for separating a vectored signal. Versions 1 and 2 have inputs on 0.2-inch centers and versions 3 and 4 have inputs on 0.1-inch centers.

You can use a MERGE symbol as a “demerger” by using a different version of the symbol. Versions 2 and 4 of each MERGE symbol are used to separate a vectored signal into several signals.

Note: Concept HDL processes designs faster if you use BIT TAP symbols instead of MERGE symbols to slice signals. For more information on BIT TAP symbols, see BIT TAP on page 618.

You can also create new MERGE symbols by copying the `HDL_CONCAT=TRUE` property from one of the MERGE symbols in the Standard library to the new symbol.

Using Merge Symbols

To use a MERGE symbol to merge signals

1. In Concept HDL, choose *Component > Add*.

   The *Component Browser* appears.

2. Select `standard` from the *Library* drop-down list.

3. Select a MERGE symbol from the Cells list. The component is attached to your cursor.

   To merge two signals, select the 2 MERGE symbol. To merge three signals, select the 3 MERGE symbol, and so on.

4. Move the cursor to the Concept HDL drawing area, but do not click in it.

5. Select version 1 or 3 of the MERGE symbol.

   To select the Version, click the right mouse button and choose *Version*. The next version of the component is displayed. Repeat this step until the version you want is displayed.

   Note: Versions 1 and 3 of a MERGE symbol are for merging signals; versions 2 and 4 are for separating vectored signals.

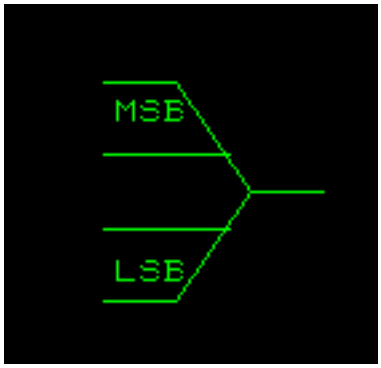
6. Click in the Concept HDL drawing area to place the symbol.

7. Close the *Component Browser*.

8. Connect the signals you want to merge to the left pins of the MERGE symbol.

Example

Version 3 of the 4 MERGE symbol below is used to merge four signals:



To use a MERGE symbol to separate a vectored signal

1. In Concept HDL, choose *Component* > *Add*.

   The *Component Browser* appears.

2. Select `standard` from the *Library* drop-down list.

3. Select a MERGE symbol from the Cells list. The component is attached to your cursor.

   To separate a vectored signal into two signals, select the 2 MERGE symbol. To separate a vectored signal into three signals, select the 3 MERGE symbol, and so on.

4. Move the cursor to the Concept HDL drawing area, but do not click in it.

5. Select version 2 or 4 of the MERGE symbol.

   To select the Version, click the right mouse button and choose *Version*. The next version of the component is displayed. Repeat this step until the version you want is displayed.

   Note: Versions 2 and 4 are for separating vectored signals; versions 1 and 3 are for merging signals.

6. Click in the Concept HDL drawing area to place the symbol.

7. Close the *Component Browser*.

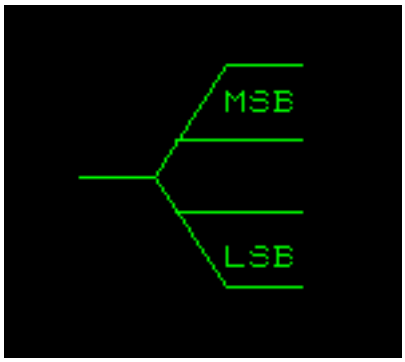
8. Connect the signal you want to separate to the left pin of the MERGE symbol.

9. Name the output signals.

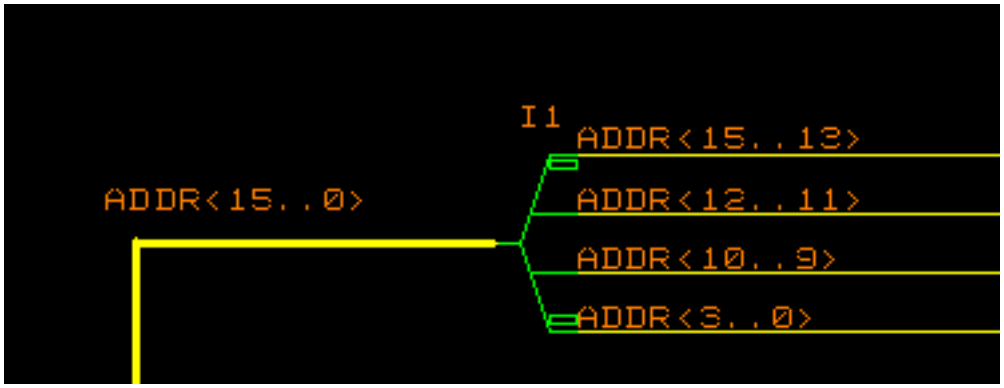
   For example, to separate a bus `addr<15..0>` into four signals, name the output signals `addr<8..0>`, `addr<10..9>`, `addr<12..11>`, and `addr<15..13>`.

Example

Version 4 of the 4 MERGE symbol below is used to separate a vectored signal into four signals.



The following example shows how the 4 MERGE symbol given above is used to separate a vectored signal into many signals.



Rules for Using MERGE symbols

- The width of the output net must match the sum of the widths of the input nets. The following assumptions apply:
  - If the output is not named (that is, its width is not specified), the width of the output net is assumed to be the sum of the widths of the input signals.
  - If some input nets are not named (that is, their width is not specified), each unnamed input net is assumed to be 1-bit wide.
- If the input nets are unnamed and the output net is also unnamed, you must specify either the widths of all the input nets or the width of the output net. You can use a SLASH symbol to specify the width of an unnamed signal. For more information on the SLASH symbol, see Specifying the Size of Nets on page 135.

- If one or more input pins are unconnected, the width of the output net must be greater than or equal to the sum of the input widths, assuming a width of 1 for each unconnected pin.
- The name of the output pin must be the highest alphanumeric value of all the pins on the symbol. For example, if the input pins are named AA, DD, and FF, the output pin cannot be named BB.
- Versions 3 and 4 of 2 MERGE, 4 MERGE, 6 MERGE, 8 MERGE, and 10 MERGE have off-grid outputs. Do not use them if you need to have all pins on the grid.

Signal Slices (Bit and Part Selects)

In VHDL, a slice is a way to reference specific bits of a vectored signal, port, or signal alias. In Verilog, slices are called “bits” and “part selects.”

You can create slices in Concept HDL schematics either textually or graphically (using the SLICE symbol in the Standard library). You can also use the `tap` command in the Concept-HDL Console window to create slices.

To create a slice textually

➤ Specify the bits you want to slice in the signal name.

For example, if you have a signal `addr<31:0>`, and you want to reference its leftmost bit, attach the signal name `addr<31>` to a wire or a pin. If you want to reference the leftmost three bits, attach the signal name `addr<31:29>` to a wire or a pin.

Note: If you do not select the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box, you must be careful about the syntax you use to specify the width of the signal. See Naming Signals on page 126 for more information.

To create a slice graphically

1. In Concept HDL, choose *Component* > *Add*.

   The *Component Browser* appears.

2. Select `standard` from the *Library* drop-down.
3. Select `SLICE` from the *Cells* list.
4. Click in the Concept HDL drawing area to place the symbol.

   If you want to create several slices, continue clicking until all the slice symbols are placed.

If you placed a single SLICE symbol,

a. From the *Text* menu, choose *Attributes*.

b. Click on the edge of the straight part of the SLICE symbol to display the *Attributes* dialog box.

c. Change the value of the `BN` property to the bit number you want to tap.

d. Click OK to save the changes and close the *Attributes* dialog box.

Note: The value of the `BN` property can be a range specification to tap multiple bits.

If you added more than one SLICE symbol,

a. In Concept HDL, choose *Wire > Bus Tap Values*.

The *Bus Tap Range* dialog box appears.

b. Specify the Most Significant Bit (*MSB*), Least Significant Bit (*LSB*), and the increment between them.

c. Click *OK*.

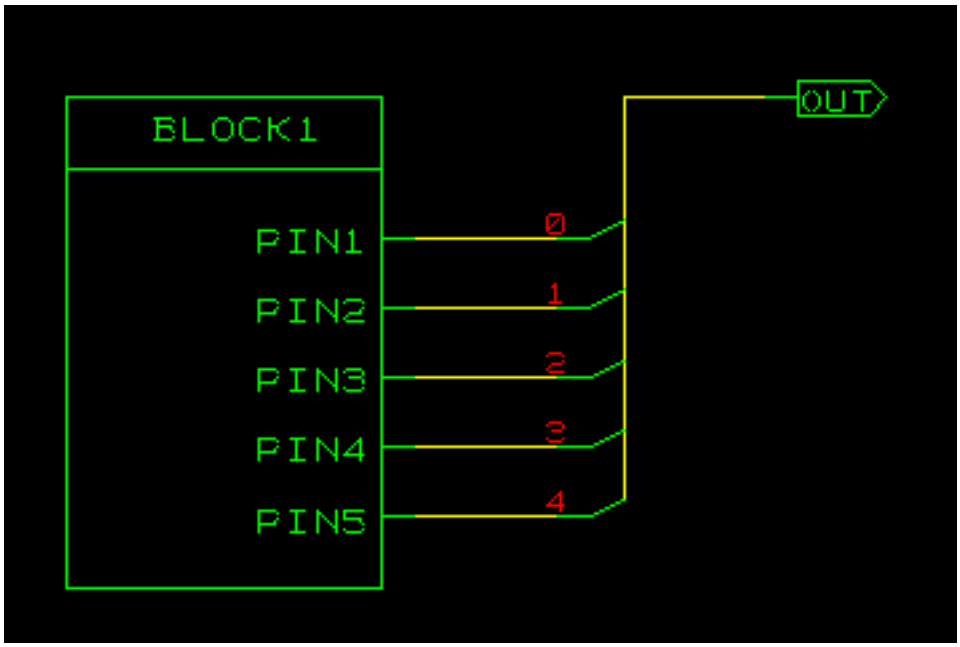
d. Draw a line across the SLICE symbols you placed on the schematic. The slices get numbered from the MSB to the LSB.

5. Wire the bent part of the SLICE symbols to a vectored signal and the straight part to the pin to which you want to connect the SLICE.

Typically, you do not name the wire on the straight part of the SLICE because the SLICE provides the name for the wire.

Example

In this example, five SLICE symbols were placed on the schematic, and the *Wire* > *Bustap Values* menu option in Concept HDL was used to number the `BN` property on each SLICE.



To slice multiple bits

1. Add a SLICE symbol from the Standard library.
2. From the *Text* menu, choose *Attributes*.
3. Click on the edge of the straight part of the SLICE symbol to display the *Attributes* dialog box.
4. Change the value of the `BN` property to a range specification. For example, you can set the value to `1 to 10` or `10:1` or `size:1`.

Rules for Using SLICE Symbols

- You cannot slice a concatenation of signals.
- You can set the value of the `BN` property on a SLICE to a range specification, for example, `BN=1 to 10`.

Setting the Verilog Logic Type for Ports and Signals

The default Verilog logic type for ports and signals is `WIRE`. You can change the default Verilog logic type for a project. Examples of other Verilog types for ports and signals include `WAND` and `WOR`.

Note: Verilog does not support the use of abstract data types such as floating points and integers.

The Verilog logic type is determined by the `VLOG_NET_TYPE` property. With this property, you can choose the Verilog logic type at the following levels:

- Setting the Verilog Logic Type for All Ports and Signals in All Drawings of a Project
- Setting the Verilog Logic Type for All Ports and Signals in a Drawing
- Setting the Verilog Logic Type for a Specific Port
- Setting the Verilog Logic Type for a Specific Signal

The Verilog logic type you select for an individual port or signal has precedence over the logic type you specify for the drawing, which in turn has precedence over the logic type you set for the project.

Note: Verilog allows a signal of type `WIRE` to be connected to ports on instances of different types.

Setting the Verilog Logic Type for All Ports and Signals in All Drawings of a Project

You can specify the default Verilog logic type for all the ports and signals in all drawings of a project. You can change these defaults for a project.

You can override the default Verilog logic type specified for the project by specifying the Verilog logic type for all ports and signals in a specific drawing. You can further override the Verilog logic type specified for a drawing by specifying the Verilog logic type for individual ports and signals.

To set the default Verilog logic type for a project

1. In Concept HDL, choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab.

   Ensure that the *Create Netlist* check box is selected.

3. Click the *Options* button next to the *Verilog* check box.

   The *Verilog Netlist* dialog box appears.

4. In the *Default Net Type* text box, type the Verilog logic type you want to use for all the ports and signals in the design. The default type is `WIRE`. Examples of other Verilog types for ports and signals include `WAND` and `WOR`.

5. Click *OK* to save the changes.

6. Click *OK* to close the *Concept Options* dialog box.

Setting the Verilog Logic Type for All Ports and Signals in a Drawing

You can override the default Verilog logic type specified for the project by specifying the Verilog logic type for all ports and signals in a specific drawing. Specify the Verilog logic type for all ports and signals in a drawing by attaching the `VLOG_NET_TYPE` property to a VERILOG_DECS symbol. For more information on the VERILOG_DECS symbol, see VHDL_DECS and VERILOG_DECS Symbols on page 613.

To set the Verilog logic type for all ports and signals in a drawing

1. Add a VERILOG_DECS symbol from the Standard library to the first page of the drawing.

2. Choose *Text* > *Attributes* and click on the VERILOG_DECS symbol to display the *Attributes* dialog box.

3. Click *Add*.

4. Type `VLOG_NET_TYPE` in the *Name* text box and type the Verilog logic type in the *Value* text box.

   The value of the `VLOG_NET_TYPE` property can be `WIRE`, `WAND`, `WOR`, or any other legal Verilog type.

5. Click *OK* to save the changes and close the *Attributes* dialog box.

Note: The `VLOG_NET_TYPE` property on an individual port or signal has precedence over the `VLOG_NET_TYPE` property on a VERILOG_DECS symbol, which in turn has precedence over the Verilog logic type specified for the project.

Setting the Verilog Logic Type for a Specific Port

You can set the Verilog logic type for each port individually or as a default for all the ports of a symbol. The type is declared with the VLOG_NET_TYPE property. Because Verilog does not support abstract data types, the Verilog logic type of ports cannot be an abstract data type.

To declare the Verilog logic type of a port

1. In Concept HDL, choose *Text > Attributes.*
2. Click on a pin on the port to display the *Attributes* dialog box.
3. Click *Add*.
4. Type VLOG_NET_TYPE in the *Name* text box and type the Verilog logic type in the *Value* text box.

   The value of the VLOG_NET_TYPE property can be WIRE, WAND, WOR, or any other legal Verilog type.
5. Click *OK* to save the changes and close the *Attributes* dialog box.

Note: The VLOG_NET_TYPE property on an individual port has precedence over the VLOG_NET_TYPE property on a VERILOG_DECS symbol, which in turn has precedence over the Verilog logic type specified for the project.

Setting the Verilog Logic Type for a Specific Signal

You can set the Verilog logic type for each signal. The type is declared with the VLOG_NET_TYPE property.

To set the Verilog logic type for a signal

1. In Concept HDL, choose *Text > Attributes.*
2. Click on the signal to display the *Attributes* dialog box.
3. Click *Add*.
4. Type VLOG_NET_TYPE in the *Name* text box and type the Verilog logic type in the *Value* text box.

   The value of the VLOG_NET_TYPE property can be WIRE, WAND, WOR, or any other legal Verilog type.
5. Click *OK* to save the changes and close the *Attributes* dialog box.

Note: The VLOG_NET_TYPE property on an individual signal has precedence over the VLOG_NET_TYPE property on a VERILOG_DECS symbol, which in turn has precedence over the Verilog logic type specified for the project.

To assign a type Supply 0 or Supply 1 to power and ground symbols

1. In Concept HDL, choose *Text > Attributes*.
2. Click on the symbol or on a pin of the symbol to display the *Attributes* dialog box.
3. Click *Add*.
4. Type VLOG_NET_TYPE in the *Name* text box and type the Verilog logic type in the *Value* text box.

   The value of the VLOG_NET_TYPE property can be WIRE, WAND, WOR, or any other legal Verilog type.
5. Click *OK* to save the changes and close the *Attributes* dialog box.

For VHDL, the VHDL_INIT property should be attached to the power or ground symbol or to its pin with a value of 1 or 0 respectively. This results in the power signal getting assigned that value in the VHDL created by Concept HDL.

Setting the VHDL Logic Type for Ports and Signals

The default VHDL logic type for all ports and signals in Concept HDL schematics is STD_LOGIC (for scalar ports and signals) and STD_LOGIC_VECTOR (for vectored ports and signals). You can change these defaults for a project.

Examples of other VHDL logic types you can use for ports and signals include BIT and BIT_VECTOR. VHDL also lets you declare a signal or port as an abstract data type such as a floating point number or integer. See Abstract Data Types in VHDL on page 167 for more information about abstract data types and the restrictions on their use.

The VHDL type of a port or signal is determined by the VHDL_SCALAR_TYPE and VHDL_VECTOR_TYPE properties. With these properties, you can choose the VHDL logic type for ports and signals at the following levels:

- Setting the VHDL Logic Type for All Ports and Signals in All Drawings of a Project
- Setting the VHDL Logic Type for All Ports and Signals in a Drawing
- Setting the VHDL Logic Type for a Specific Port
- Setting the VHDL Logic Type for a Specific Signal

The VHDL logic type you select for an individual port or signal has precedence over the logic type you specify for a drawing, which in turn has precedence over the logic type you set for the project.

Note: You can also set the initial value of a signal with the `VHDL_INIT` property. For more information, see Setting the Initial Value of a Signal on page 135.

Setting the VHDL Logic Type for All Ports and Signals in All Drawings of a Project

You can specify the default VHDL logic type for all the ports and signals in all drawings of a project. You can change these defaults for a project.

You can override the default VHDL logic type specified for the project by specifying the VHDL logic type for all ports and signals in a specific drawing. You can further override the VHDL logic type specified for a drawing by specifying the VHDL logic type for individual ports and signals.

To set the VHDL logic type for a project

1. In Concept HDL, choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab.

   Ensure that the *Create Netlist* check box is selected.

3. Click the *Options* button next to the *VHDL* check box.

   The *VHDL Netlist* dialog box appears.

4. In the *Vector Type* text box, enter the VHDL logic type you want to use for the vectored ports and signals in the design. The default type is `STD_LOGIC_VECTOR`.

5. In the *Scalar Type* text box, enter the VHDL logic type you want to use for the scalar ports and signals in the design. The default type is `STD_LOGIC`.

6. Click *OK* to save the changes.

7. Click *OK* to close the *Concept Options* dialog box.

Setting the VHDL Logic Type for All Ports and Signals in a Drawing

You can override the default VHDL logic type specified for the project by specifying the VHDL logic type for all ports and signals in a specific drawing. Specify the VHDL logic type for all the ports and signals in a drawing by attaching the `VHDL_SCALAR_TYPE` and

VHDL_VECTOR_TYPE properties to a VHDL_DECS symbol. For more information on the VHDL_DECS symbol, see VHDL_DECS and VERILOG_DECS Symbols on page 613.

To set the VHDL type for a drawing

1. Add a VHDL_DECS symbol to the first page of the schematic.
2. Choose *Text > Attributes*.
3. Click on the VHDL_DECS symbol to display the *Attributes* dialog box.
4. Add the VHDL_SCALAR_TYPE and VHDL_VECTOR_TYPE properties as below:
   - ❑ Add the VHDL_SCALAR_TYPE property if the drawing has only scalar ports.
   - ❑ Add the VHDL_VECTOR_TYPE property if the drawing has only vectored ports.
   - ❑ Add the VHDL_SCALAR_TYPE and VHDL_VECTOR_TYPE properties if the drawing has both scalar and vectored ports.

   The value of the VHDL_SCALAR_TYPE property can be STD_LOGIC, BIT, or any other legal VHDL scalar type. The value of the VHDL_VECTOR_TYPE property can be STD_LOGIC_VECTOR, BIT_VECTOR, or any other legal VHDL vector type.
5. Click *OK* to save the changes and to close the *Attributes* dialog box.

Note: The VHDL_SCALAR_TYPE or VHDL_VECTOR_TYPE property on an individual port or signal has precedence over the property on a VHDL_DECS symbol, which in turn has precedence over the VHDL logic type specified for the project.

Setting the VHDL Logic Type for a Specific Port

You can set the VHDL logic type for each port individually or as a default for all the ports of a symbol. The type is declared with the VHDL_SCALAR_TYPE and VHDL_VECTOR_TYPE properties.

The VHDL logic type can be an abstract data type. See Abstract Data Types in VHDL on page 167 for more information about abstract data types and the restrictions on their use.

Note: The VHDL_SCALAR_TYPE or VHDL_VECTOR_TYPE property on an individual port has precedence over the property on a VHDL_DECS symbol, which in turn has precedence over the VHDL logic type specified for the project.

To declare the VHDL logic type of a port

1. In Concept HDL, choose *Text > Attributes.*

2. Click on a pin on the port to display the *Attributes* dialog box.
3. Add the VHDL_SCALAR_TYPE and VHDL_VECTOR_TYPE properties as below:
   - If the symbol has only scalar ports, add the VHDL_SCALAR_TYPE property.
   - If the symbol has only vectored ports, add the VHDL_VECTOR_TYPE property.
   - If the symbol has both vectored and scalar ports, add both the VHLD_SCALAR_TYPE and VHDL_VECTOR_TYPE properties.

   The value of the VHDL_SCALAR_TYPE property can be STD_LOGIC, BIT, or any other legal VHDL scalar type. The value of the VHDL_VECTOR_TYPE property can be STD_LOGIC_VECTOR, BIT_VECTOR, or any other legal VHDL vector type.
4. Click *OK* to save the changes and to close the *Attributes* dialog box.

Note: The VHDL_SCALAR_TYPE or VHDL_VECTOR_TYPE properties attached to pins of ports have precedence over the VHDL_SCALAR_TYPE or VHDL_VECTOR_TYPE properties attached to the origin of the symbol.

Setting the VHDL Logic Type for a Specific Signal

To set the VHDL logic type for a signal

1. In Concept HDL, choose *Text > Attributes.*
2. Click on the signal to display the *Attributes* dialog box.
3. Add the VHDL_SCALAR_TYPE and VHDL_VECTOR_TYPE properties as below:
   - If the signal is scalar, add the VHDL_SCALAR_TYPE property.
   - If the signal is vectored (a bus), add the VHDL_VECTOR_TYPE property.

   The value of the VHDL_SCALAR_TYPE property can be STD_LOGIC, BIT, or any other legal VHDL scalar type. The value of the VHDL_VECTOR_TYPE property can be STD_LOGIC_VECTOR, BIT_VECTOR, or any other legal VHDL vector type.
4. Click *OK* to save the changes and to close the *Attributes* dialog box.

Specifying Ranges for Ports, Signals and Aliases

Range specifications are used in VHDL and Verilog to declare the widths of vectored ports, signals, and aliases and to create slices of these objects. When you create Concept HDL schematics, you can use VHDL or Verilog syntax to specify a range.

Follow these conventions for range specifications:

- Enclose range specifications in angle brackets <>.

  Note: You can also use ( ) or [ ] if you select the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box.

- Specify a descending range with a colon (:), ellipsis (..), or either of the strings `downto` or `DOWNTO`.

  Note: If the left and right bounds of a range specification are constant integers and the right integer is greater than the left, the colon (:) is interpreted as specifying an ascending range.

- Specify ascending ranges with a colon (:), ellipses (..), or either of the strings `to` or `TO`.

Examples

The following are examples of legal range specifications in Concept HDL schematics:

<10 downto 0>	11-bit descending range
<10..0>	11-bit descending range
<0 to 10>	11-bit ascending range
<10:0>	Colon (:) is the same as *downto*
<0:10>	Colon (:) also works like *to*
<size-1:0>	Parameterized descending range
<0 to size-1>	Parameterized ascending range

The following examples are illegal in Concept HDL schematics
:

<10 to 0>	Illegal. `to` must be an ascending range.
<0 downto 10>	Illegal. `downto` must be a descending range

If you specify the port range in a user-defined package, you should ensure that the width of the ports in the symbol match with the port range specified in the package.

Unconstrained Ranges for Ports, Signals, and Aliases

While VHDL lets you create arrayed objects with unconstrained bounds, Concept HDL does not support unconstrained ports, signals, or aliases. If you want an unconstrained range for an object, use a parameterized range for the object.

Example

For example, in VHDL you may have the following port (a vectored port with an unconstrained range) on an entity:

```
entity CPU is
    port (
        io_addr: out std_logic_vector;
        io_busy: in std_logic;
    );
end cpu;
```

For Concept HDL, you can change the above entity declaration to the following:

```
entity CPU is
    generic (size: positive);
    port (
        io_addr: out std_logic_vector (size - 1 downto 0);
        io_busy: in std_logic;
    );
end cpu;
```

Architectures that instantiate this `CPU` entity can generate the value for the `SIZE` property by using the pre-defined `LENGTH` attribute for the signal attached to the `io_addr` port.

Similarly, if you want an unconstrained range for a signal, `ADDR`, in a schematic, declare it as a parameterized range as below:

```
ADDR (size-1:0)
```

Resolved Types and Resolution Functions

In VHDL, if a signal has multiple drivers, you must define a resolution function to resolve the signal conflict. You can declare a resolved signal in two ways in VHDL.

- The signal declaration can refer to a resolved type.
- The signal declaration can specify a resolution function and an unresolved type.

However, in Concept HDL, only the first option is possible. If you want to use a resolved signal, you must reference an existing resolved type when you declare the signal type.

Type Conversion

If you need to connect a signal of one type to a port of another type, use a type conversion function, that will result in the correct VHDL output. There are some restrictions on using type conversion functions with abstract data types.

Note: Verilog does not use type conversion functions; in Verilog, you can connect a signal of the type `wire` to ports of other types. While generating Verilog text, Concept HDL ignores the `VHDL_IN_CONVERT` and `VHDL_OUT_CONVERT` properties that are used on the schematic.

To specify type conversion functions in a schematic

- If the port is an input, attach this property to the pin:

  `VHDL_IN_CONVERT = function_name`

- If the port is an output or a buffer, attach this property to the pin:

  `VHDL_OUT_CONVERT = function_name`

- If the port is bi-directional, attach these properties to the pin:

  `VHDL_IN_CONVERT = function_name1`
  `VHDL_OUT_CONVERT = function_name2`

  For example, if a pin `IO1` in a schematic is connected to a signal `INT` and has the following properties:

  ```
  VHDL_IN_CONVERT=MYLIB.PKG.FIN
  VHDL_OUT_CONVERT=MYLIB.PKG.FOUT
  ```

  the VHDL text generated for the instance of the symbol has the following in its port map clause:

  ```
  MYLIB.PKG.FOUT(IO1) => MYLIB.PKG.FIN(INT)
  ```

Note: Attach the `VHDL_IN_CONVERT` and `VHDL_OUT_CONVERT` properties to pins of component instances in your schematic, not to the INPORT or OUTPORT port symbols.

Restrictions on Using Type Conversion Functions

If you have an object (for example, a signal) of an abstract data type connected to an object (for example, a port) of a bit-oriented type, do not specify a type conversion function using the `VHDL_IN_CONVERT` and `VHDL_OUT_CONVERT` properties. Instead, create an entity that has two ports, one of each type. This entity performs the type conversion.

However, if both objects are non-vectored types, or both objects are vectored types and have the same number of elements, you can use the `VHDL_IN_CONVERT` and `VHDL_OUT_CONVERT` properties.

Abstract Data Types in VHDL

VHDL supports a variety of data types. For example, it is possible in VHDL to have a signal or a port declared as an abstract data type. Examples of an abstract data type include a floating-point number, an integer, or a record made up of a set of data types. These abstract types are different from types such as `BIT`, `BIT_VECTOR`, `STD_LOGIC`, and `STD_LOGIC_VECTOR` because their correspondence to the hardware implementation is not explicitly stated.

Concept HDL supports the use of abstract data types through the `VHDL_SCALAR_TYPE` and `VHDL_VECTOR_TYPE` properties on schematics and by referencing these types within entity declarations.

Note: Verilog does not support the use of abstract data types. If you are using Verilog, do not use abstract data types.

Restrictions on the Use of Abstract Data Types

A signal of one type can be connected to a port of another type if a type conversion function is specified with the `VHDL_IN_CONVERT` and `VHDL_OUT_CONVERT` properties. However, if you have an object (for example, a signal) that is an abstract data type connected to another object (for example, a port on a component) that is a bit-oriented type, you should not use a type conversion function. Instead, create an entity that has two ports, one of each type. This entity performs the type conversion.

However, the `VHDL_IN_CONVERT` and `VHDL_OUT_CONVERT` properties work correctly if both objects are non-vectored types or both objects are vectored types and have the same number of elements.

Using Iterated Instances

If you use iterated instances, you can replicate parts without building them as parameterized models. When you create an iterated instance, Concept HDL automatically expands the iterated instance into multiple instances when it generates VHDL and Verilog representations of the schematic. The number of instances you want to generate must always be a constant; it cannot be a parameter.

You cannot have iterated instances of parts that have parameterized port widths. All signals attached to an iterated instance must have a fixed width. If the width of a signal matches the width of the pin, every generated instance has that signal attached. If the width of a signal is greater than the width of the pin to which it is attached, Concept HDL automatically attaches the correct bits of the signal to the correct pins.

To use the iterated instance feature, add a PATH property to an instance. The PATH property specifies the number of instances you want. For example, to generate 16 instances of a part, add the following property to the part:

```
PATH = I3<15..0>
```

The generated instance labels in Verilog and VHDL are:

```
I3_GEN_15
I3_GEN_14
I3_GEN_13
...
I3_GEN_0
```

If you have selected the *Multi-format Vectors* option in the *General* tab of the *Concept Options* dialog box, you can also set the PATH properties in one of the following three ways:

- `PATH = I3(15..0)`
- `PATH = I3[15..0]`
- `PATH = I3[15:0]`

X-Replication

X-replication, also called size replication, is a powerful mechanism for creating concise representations of a design. X-replication uses the SIZE property to duplicate parts and specify how they will be connected. To use X-replication, you must add a DEFINE symbol to the design and at least one net on the design must have an X variable. The DEFINE symbol must have the X_FIRST, X_STEP, and X_SIZE properties on it.

The number of times the design is instantiated depends on the values of the X_SIZE, X_FIRST and X_STEP properties and on the expression in which the X variable is used. In the first instance of the design, the value of X is X_FIRST. In the second instance, the value of X is X_FIRST+X_STEP, and so on until X is less than or equal to the value of X_SIZE. The number of times X can be evaluated this way is the number of times the design is instantiated.

Example

The design MYPART has a `net A(X)` and a DEFINE body with the following properties:

```
X_FIRST = 1, X_STEP = 8, and X_SIZE=32
```

The value of X in the first instance is 1, in the second 9, in the third 17, and in the fourth 25. Therefore, when MYPART is used, it is instantiated four times.

Using X-Replication

1. Add a DEFINE symbol to the schematic. The DEFINE symbol must have the following three properties on it: `X_SIZE`, `X_FIRST`, and `X_STEP`.
2. Specify the values of these properties. The values must be constants; they cannot be variables.

Rules for Using X-Replication

- The design must have a DEFINE symbol and at least one net on the design must have the X variable.
- There can be only one DEFINE symbol in a drawing.
- The DEFINE symbol must have these three properties: `X_SIZE`, `X_FIRST`, and `X_STEP`.
- The values of the `X_SIZE`, `X_FIRST`, and `X_STEP` properties must be constants; they cannot be variables.

Saving a Design

Concept HDL displays a * sign in the title bar to show that the current page needs to be saved. In the hierarchy mode, you see [*in hierarchy]**, and in the expanded mode, you see *[needs expansion]**.

Note: In the occurrence edit mode, Concept HDL does not display * sign to indicate that the drawing needs to be saved.

When you save the design, Concept HDL writes the current design on the disk. The * sign disappears from the title bar.

In the expanded mode, if you change any drawing of the expanded design, Concept HDL displays *needs expansion* on the title bar of the page. This signifies that the design needs to be re-expanded. In the occurrence edit mode, if you make any change and then come back to the expanded mode, Concept HDL displays *needs expansion* on the title bar. If you re-expand the design, the message on the title bar changes to *expanded*.

Note: When you package the design using *Packager-XL, Packager-XL* adds certain soft properties to the design, but Concept HDL does not display *needs expansion* on the title bar.

Before saving the design, Concept HDL automatically runs all the checks that are normally run when you choose *Tools > Check*. Concept HDL also checks for connectivity errors on other pages in the design.

To save an existing drawing:

➤ Choose *File > Save*.

To save an existing drawing with a new name:

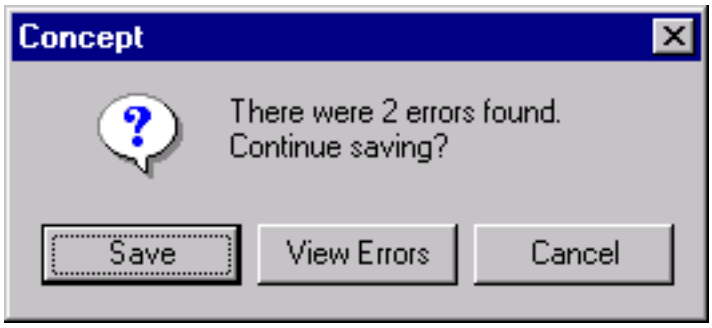
1. Choose *File > Save As*.
2. In the *View Save As* dialog box that appears, highlight the existing drawing name in the *Cell* box, and type a new drawing name.
3. Click *Save*.

To save a new drawing:

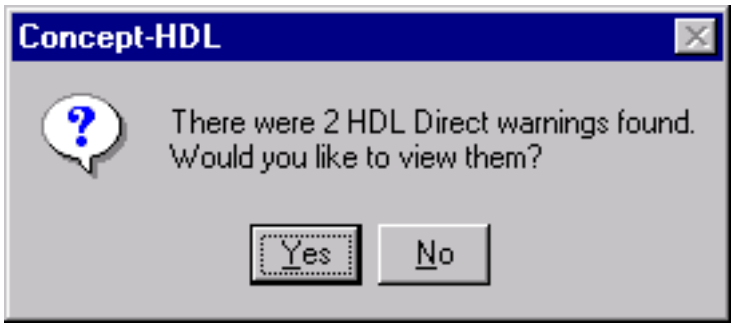
1. Choose *File > Save As*.
2. In the *View Save As* dialog box that appears, type a drawing name in the *Cell* box.

   Concept HDL appends .SCH.1.1 to the file name that you specify. For example, if the file name you enter is MEMORY, Concept HDL names the drawing MEMORY.SCH.1.1. (Concept HDL assumes version 1 and page 1 of the drawing.)
3. Click *Save*.

If any errors are found on the current page, Concept HDL reports them.



Concept HDL checks for connectivity errors on all the pages in the design and reports errors, if any.



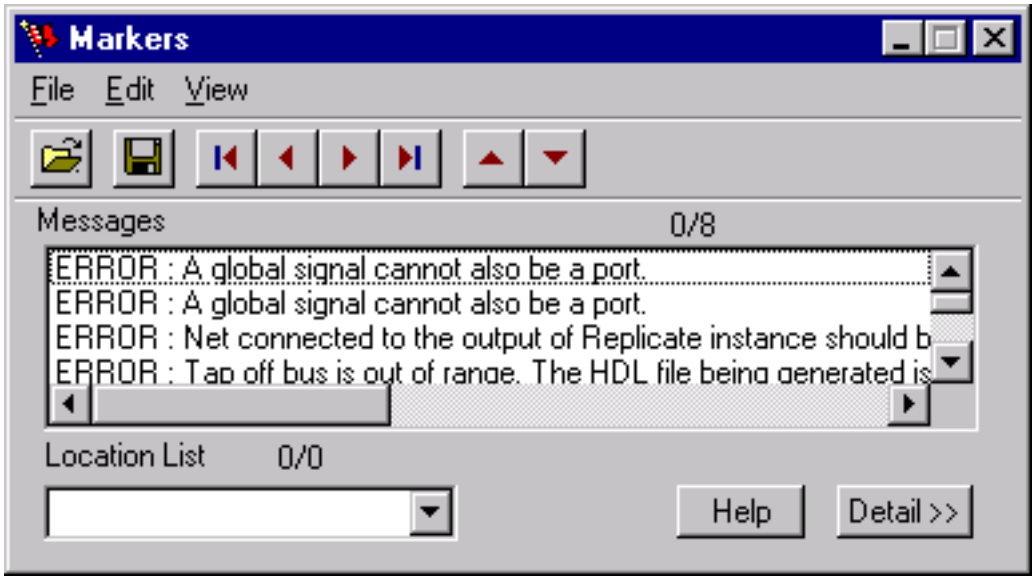
To view the errors, open the Markers control window

➤ Choose *Tools* > *Markers*.

or

➤ Click the Markers Controls button in the *Markers* toolbar.

Click Yes to view the HDL-Direct errors in the Markers window.


Markers
File Edit View
Messages 0/8
ERROR : A global signal cannot also be a port.
ERROR : A global signal cannot also be a port.
ERROR : Net connected to the output of Replicate instance should b
ERROR : Tap off bus is out of range. The HDL file being generated is
Location List 0/0
Help
Detail >>

Click on an error to find it on the schematic. Concept HDL highlights the area where the error occurred.

When you save a schematic, Concept HDL does not copy over the parts used in the schematic to the local database. Concept HDL is a by-reference editor that references all parts in the schematic from various libraries that reside at the reference or local area.

Working With Existing Designs

This section describes the following:

- Opening a Drawing on page 172
- Recovering a Drawing on page 172
- Reverting to the Previous Saved Version of a Drawing on page 173

Opening a Drawing

1. Choose *File > Open*.
2. Select a cell you want to open.
3. Click on the cell to expand. The expanded cell displays all the views in it.
4. Select the view you want to open and click *Open*. You can also double-click on the schematic view (sch_n) to view the pages and double-click on a page to open it in Concept HDL.

   Concept HDL opens schematic and symbol files. Other views (Verilog and VHDL) are opened based on the editor registered for these views in Project Manager.

Note: Plus signs (+) indicate there are lower level listings. The + changes to a minus (-) when there are no more lower level listings. For schematics, you choose the page number of the schematic that you want to open.

You can also start Concept HDL from the Project Manager to access your drawings. See the Project Manager online help for instructions on starting Concept HDL from that entry point. The steps shown here for opening a drawing still apply.

Recovering a Drawing

To recover drawings that were being edited when Concept HDL or your system crashes, do the following:

1. Choose *File > Recover*.

2. In the file browser that appears, navigate to the `./temp/xxxnedtmp` directory where Concept HDL places undo log files.

   Everytime you start Concept HDL, a temporary directory is created in the `<project_directory>/temp` directory. An undo log file for each drawing is stored in this directory. By default, `xxnedtmp` is the name of the temporary directory. If the `xxnedtmp` directory already exists, a `xxnedtmp1` directory is created. If these two directories already exist, `xxnedtmp2` is created, and so on. The name of the undo log file for the first drawing edited is `undo1.log`. The second drawing's undo log file is `undo2.log`, and so on.

3. Select the undo log file for the drawing you want to recover and click *Open*.

   Concept HDL gives the recovered drawing a unique name (for example, `RECOVER1.SCH.1.1)`. The recovered drawing is only saved in memory, not on disk.

4. Choose *File > Save As* to save the drawing with a different name.

Reverting to the Previous Saved Version of a Drawing

➤ Choose *File > Revert*.

Concept HDL displays the last saved version of the current drawing.

8

Working with Libraries and Components

Concept HDL includes extensive analog and digital libraries, and simulation models that you can use on your schematic pages. These libraries support design entry, simulation, timing, test and physical layout—a complete solution for designing digital, analog and mixed signal systems.

- For more information on digital libraries and simulation models, see the *Concept HDL Libraries Reference*.
- For more information on analog libraries and simulation models, see the *Analog Workbench* or the *PSpice* documentation.

About the Standard Library

Cadence provides a Concept HDL library of standard components that lets you define and control signals in designs. These components include merge bodies for merging signals and tap bodies for tapping bits from buses. Other special parts contained in the Standard Library are NOT bodies and differently sized drawing borders.

Although the components in the Standard Library can be used for any of the supported design types, many of them are created especially for structured designs. For more information, see Appendix C, “Using the Standard Library Symbols.”

Working with Libraries

This section describes the procedures for working with libraries.

- Adding New Libraries on page 176
- Browsing Libraries on page 176
- Adding Libraries to the Search Stack on page 176
- Removing Libraries from the Search Stack on page 177

■ Defining Library Search Order on page 177

Adding New Libraries

Available libraries are defined in the `cds.lib` file. To add new libraries, you must edit the `cds.lib` file. Do this using *Setup* in the Project Manager.

See the Project Manager Online Help for more information.

Browsing Libraries

1. Choose *File > View Search Stack*.

   The *Search Stack* dialog box appears, showing the list of active libraries.

2. Select a library and double-click on it, or click *Browse*.

The *Component Browser* dialog box displays the list of components for the library you specify.

Adding Libraries to the Search Stack

1. Choose *File > View Search Stack*.

   The *Search Stack* dialog box appears, showing the list of active libraries.

2. Click *Edit >>*.

   The *Search Stack* dialog box expands to display the libraries installed in your `cds.lib` file.

3. Select a library from the list of available libraries on the right.

4. Optionally, specify *Top* or *Bottom* in the *Position* box to tell Concept HDL where to place the library in the active libraries list.

5. Click *< Add.*

   The library you specify is added to the list of active libraries.

6. Click *<< Done*.

Removing Libraries from the Search Stack

1. Choose *File > View Search Stack*.

   The *Search Stack* dialog box appears, showing a list of active libraries.

2. Click *Edit >>*.

   The *Search Stack* dialog box expands to display the libraries installed in your `cds.lib` file.

3. Select a library in the active libraries list on the left and click *Ignore >.*

4. Click *Yes* in the confirmation box.

   The library is removed from the list.

Note: You can add the library back to the list of active libraries.

Defining Library Search Order

To define a library search order as you add libraries

1. Choose *File > View Search Stack*.

   The *Search Stack* dialog box appears, showing the list of active libraries.

2. Click *Edit >>*.

   The *Search Stack* dialog box expands to display the libraries installed in your `cds.lib` file.

3. Select a library.

4. Specify *Top* or *Bottom* in the *Position* box to instruct Concept HDL where to place the library in the active libraries.

5. Click *< Add*.

   The library you specify is added to the list of active libraries.

6. Click *<< Done*.

To redefine the entire library search order

1. Expand the *Search Stack* dialog box.

2. Press Ctrl + left and select each library from the active libraries list on the left.

3. Click *Ignore >*.
4. Click *Yes* in the confirmation box.
5. Press Ctrl + left and select libraries in the desired search order in the list of available libraries on the right.
6. Click *< Add*.

   Libraries are listed in the Search Stack in the order in which you have added them.
7. Click *<< Done*.

Working with Components

This section describes the procedures for working with components.

- Browsing the Component List on page 179
- Creating Concept HDL Parts on page 179
- Creating a Symbol in Concept HDL on page 180
- Creating Entity Declarations from Symbols on page 183
- Creating the chips.prt File on page 186
- Creating a Part Table File on page 187
- Adding a Component on page 188
- Modifying Components on page 189
- Replacing a Component on page 190
- Defining Physical Property Options on page 191
- Breaking Up a Component on page 194
- Changing Pin States on a Component on page 194
- Choosing a Version of a Component on page 195
- Mirroring Components or Blocks on page 195
- Changing the Orientation of Components or Text on page 195
- Sectioning a Component on page 196
- Swapping Pins on a Component on page 197

- Ways to Determine if a Component Has Bus-Through Pins on page 197
- Deleting a Library Component (Cells, Views, and Files) on page 197
- Creating a SYNONYM on page 197
- Tapping a bit with a TAP symbol on page 198

Browsing the Component List

To browse logical components one library at a time

1. Click *Component* > *Add*.

   The *Component Browser* dialog box appears.

2. Scroll the library list in the *Cells* area of the *Library View* tab.

To browse categories of logical components

1. Click *Component* > *Add*.

   The *Component Browser* dialog box appears.

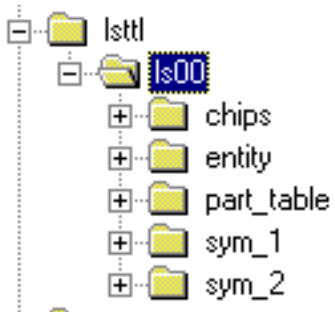
2. Scroll the category list in the *Category View* tab.

Creating Concept HDL Parts

You may follow these procedures and guidelines for creating parts if you do not have PCB Librarian installed. If you have installed PCB Librarian, you may use it to create parts for use in Concept HDL designs.

Each Concept HDL part is a collection of views. In the Lib:cell.view structure, a library and a cell (parts) are directories. Under each part, there is a directory for a view type. Each view directory contains a file that defines the view.

Given below is the directory structure that has the files and directories that define the part ls00.



In the above figure, `lsttl` is the library name, `ls00` is the part name (cell level directory), and the directories underneath contain the view files for `ls00`.

To create a Concept HDL part,

1. Create a directory with the part name `ls00`.
2. Create the following directories underneath the `ls00` directory:
   - chips
   - part_table
   - sym_1
3. Create a symbol in Concept HDL.
4. Save the symbol view files created in Concept HDL under the `sym_1` directory.
5. Create a `chips.prt` file.
6. Save the `chips.prt` file under the chips directory.
7. Create a part table file.

Creating a Symbol in Concept HDL

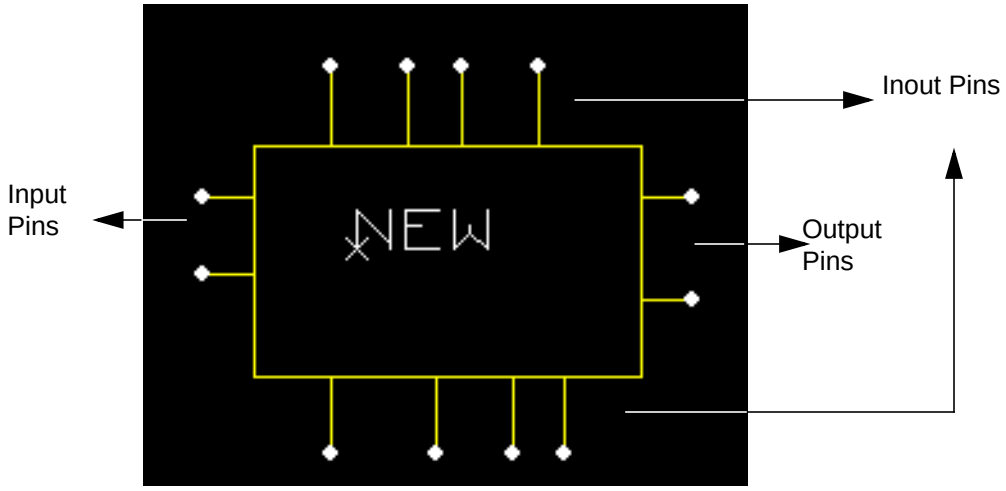
To create a new symbol, you should be in the symbol view.

1. In Concept HDL, choose *File* > *Open*.
2. From the *Library* drop-down list, select the library in which the new part is to be added.
3. In the *Cell* field, specify the new symbol name.

4. From the *View* drop-down list, select *Symbol.*
5. Specify Version as 1 and click *Open*.
6. Choose *Wire > Draw*.
7. Draw a symbol shape.
8. Choose *Wire > Draw* to draw pin stubs on the symbol.
9. Choose *Wire > Dot* for adding a pin to the symbol.

   Selecting *Wire > Dot* adds a dot to the symbol. This dot can be added on the edge of the pin stubs to represent a pin.

   Note: By default, Concept HDL treats a pin as an input, output, or an inout pin depending on which side of the symbol it is attached. By default, a pin that is on the left of a symbol is an input pin. A pin attached to the right of the symbol is an output and pins on the top and the bottom of a symbol are inout pins. You can change the default properties by adding the `vhdl_mode` or `vlog_mode` properties to the pins and assigning them the desired values. For example, to use a pin that in on the right of a symbol, as an input pin, add the `vlog_mode` property to the pin with value as `IN`.



10. The next step is to name the pin. Choose *Wire > Signal Name*.
11. In the *Signal Name* dialog box, specify the pin name and click on the dot representing the pin. The name is attached to the pin.

    Alternatively, you can add the `pin_name` property with the pin name as the property value for each pin. To do this, choose *Text > Property*. Enter `pin_name` in the *Property Name* field and the pin name in the *Property Value* field.

12. Choose *Text > Notes*.

Add pin names and attach the pin names to the respective pins.

Note: This step is required so that the pin names are visible when you instantiate the symbol in a schematic.

13. Choose *File > Save*.

Note: You can create multiple versions of a symbol. The second version of a symbol can be created only after you have created the first version of a symbol. You can create a new symbol with same name and assign the Version as 2. You can also save the existing symbol as Version 2 and then make modifications to version 2. To save the existing symbol with a different version, choose *File > Save As*. Specify the version as 2 and click *Save*.

For information on guidelines to follow while creating symbols, see *Concept HDL Libraries Reference*.

Symbol Naming Conventions

Follow the following Concept HDL rules while creating symbols:

- Symbol names must be legal Verilog and VHDL names.
- If the pins have the same base name, they are part of the same VHDL or Verilog port. For example, pins `SEL(1)` and pins `SEL(0)` represent the single port `SEL(1:0)` in the entity declaration.

  The following are some examples of pin names:

Name	Description
A	single-bit pin
`SEL(1:0)`	two-bit wide pin
`SEL(1)`	one bit of a two-bit wide vectored port
`SEL(0)`	one bit of a two-bit wide vectored port
`I(size-1:0)`	parameterized pin width
`I(SIZE-1:0)`	SCALD style parameterized pin width

Creating Entity Declarations from Symbols

If the parts you are using do not have an entity declaration in the design library, you can use Concept HDL to automatically generate entity declarations from symbols. This section describes the properties you can add to the symbol to ensure that an accurate entity declaration is generated. Typically, you make these properties invisible in the symbol view.

You can add properties for:

- Declaring VHDL Generics or Verilog Parameters
- Declaring Port Modes
- Declaring the VHDL Logic Type of Ports

  For more information, see Setting the VHDL Logic Type for Ports and Signals on page 160.
- Declaring the Verilog Logic Type of Ports

  For more information, see Setting the Verilog Logic Type for Ports and Signals on page 157.
- Declaring Libraries
- Declaring Use Clauses

Declaring VHDL Generics or Verilog Parameters

To define VHDL generics or Verilog parameters:

➤ Attach the following property to the origin of the symbol.

```
GENERICn=name:type
```

where n is a unique number, `name` is the name of the generic parameter, and `type` is the generic parameter type.

Declaring Port Modes

For every port in your symbol, attach the `VLOG_MODE` or `VHDL_MODE` property on one of the pins of the port.

Note: If a port has several pins, you need to attach the property on only one of the pins.

To declare the port mode in Verilog, attach one of the following properties:

VLOG_MODE=INPUT
VLOG_MODE=OUT
VLOG_MODE=INOUT
VLOG_MODE=BUF
VLOG_MODE=LINKAGE

To declare the port mode in VHDL, attach one of the following properties:

VHDL_MODE=IN
VHDL_MODE=OUT
VHDL_MODE=INOUT
VHDL_MODE=BUF
VHDL_MODE=LINKAGE

If you want to read the value of an OUT port inside an architecture:

- Declare the port as an INOUT.

  Or

- Use behavioral assignments.

How port modes are determined when you save a symbol

When you save a symbol, the port mode of the ports on the symbol is determined as below:

1. If the VLOG_MODE or VHDL_MODE property is attached to a port on the symbol, the value of the property is used to determine the port mode of the port.
2. If neither the VLOG_MODE nor the VHDL_MODE property is attached to a port on the symbol, the port mode for the port on the symbol will be determined from the chips.prt file as below:
   - If the BIDIRECTIONAL=TRUE property is attached to a pin, the port mode is INOUT
   - Else if the OUTPUT_TYPE property with any (or no) combination of INPUT_LOAD and OUTPUT_LOAD properties are attached to a pin, the port mode is OUTPUT
   - Else if only the INPUT_LOAD property is attached to a pin, the port mode is INPUT
   - Else if only the OUTPUT_LOAD property is attached to a pin, the port mode is OUTPUT
   - If both the INPUT_LOAD and OUTPUT_LOAD properties are attached to a pin, and the OUTPUT_TYPE or BIDIRECTIONAL=TRUE property is not attached to the pin,

the port mode, cannot be determined from the `chips.prt` file. The port mode will be determined by steps 3 or 4 below.

Note: If both the `OUTPUT_TYPE` and `BIDIRECTIONAL=TRUE` properties are attached to a pin, the `BIDIRECTIONAL` property takes precedence over the `OUTPUT_TYPE` property and the port mode is `INOUT`.

3. If the `VLOG_MODE` or `VHDL_MODE` property is not attached to ports on the symbol, and if the `chips.prt` file does not exist, the port mode of ports on the symbol will be determined from the schematic. For example, if the signal A on the schematic is connected to an `OUTPORT` symbol, the port mode for pin A on the symbol will be declared as `OUT`.
4. If neither the `VLOG_MODE` nor the `VHDL_MODE` property is attached to a port on the symbol, and if both the schematic and the `chips.prt` file do not exist, the port mode of the port on the symbol will be determined by Concept HDL using its internal algorithms.

   Note: Cadence recommends that you use the `VLOG_MODE` and `VHDL_MODE` properties to declare port modes when you are creating a symbol in Concept HDL. This ensures that the port mode of the ports on the symbol are declared as per your requirements.

Declaring Libraries

To generate library clauses from a Concept HDL symbol view:

➤ Attach the following property to the origin of the symbol:

`LIBRARY`*`n`* `= `*`libname`*

where *n* is a unique number and *libname* is the name of the library.

Example

```
LIBRARY1 = ieee
```

Declaring Use Clauses

To generate use clauses from a Concept HDL symbol drawing view:

➤ Attach the following property to the origin of the symbol:

`USEn = `*`libname`*

where *n* is a unique number and *libname* is the name of the library.

Example

```
USE1 = IEEE_VITAL_PRIMITIVES.ALL
```

Creating the chips.prt File

The `chips.prt` file is used by Packager-XL to associate pin numbers and names in your part. This file can be created using a text editor like vi or Windows Notepad. Given below is a sample `chips.prt` file with descriptions (marked #) on sections:

```
FILE_TYPE=LIBRARY_PARTS;
# This is the header. This line identifies the type of the file.
TIME=' COMPILATION ON THU JAN 10 14:52:02 1991  ';
# This is just a comment.
primitive '74LS00','74LS00_DIP';
# There could be multiple primitives. The basic primitive name in this case is
74LS00. Adding an _DIP specifies the PACK_TYPE as DIP. There are other PACK_TYPE
values like SOIC, BG, FG etc. You can specify the pin name-number assignment for
multiple primitives in one section. You need to specify a different primitive when
the pin name-number assignment is different from the basic primitive.
pin
'B'<0>:
# This is the name of the pin. In this case, B<0> represents an element of a vector
pin. All pins and the pin numbers are to be written in this file.
INPUT_LOAD='(-0.4,0.02)';
PIN_NUMBER='(13,10,5,2)';
# The 4 pin numbers represent the pin numbers in each of the 4 sections of the
device.
PIN_GROUP='1';
'A'<0>:
INPUT_LOAD='(-0.4,0.02)';
PIN_NUMBER='(12,9,4,1)';
PIN_GROUP='1';
'-Y'<0>:
OUTPUT_LOAD='(8.0,-0.4)';
PIN_NUMBER='(11,8,6,3)';
end_pin;
body
POWER_PINS='(VCC:14;GND:7)';
# Name:pin number; name2:pin_number2......
FAMILY='LSTTL';
PART_NAME='74LS00';
BODY_NAME='LS00';
DEFAULT_SIGNAL_MODEL='SN74LS00N  TI';
JEDEC_TYPE='DIP14_3';
CLASS='IC';
TECH='74LS';
end_body;
end_primitive;
# You can enter the second primitive after end_primitive.This is typically done for
different pack_types.
primitive '74LS00_SOIC';
pin
'B'<0>:
INPUT_LOAD='(-0.4,0.02)';
PIN_NUMBER='(13,10,5,2)';
PIN_GROUP='1';
```

```
'A'<0>:
INPUT_LOAD='(-0.4,0.02)';
PIN_NUMBER='(12,9,4,1)';
PIN_GROUP='1';
'-Y'<0>:
OUTPUT_LOAD='(8.0,-0.4)';
PIN_NUMBER='(11,8,6,3)';
end_pin;
body
POWER_PINS='(VCC:14;GND:7)';
FAMILY='LSTTL';
PART_NAME='74LS00';
BODY_NAME='LS00';
DEFAULT_SIGNAL_MODEL='SN74LS00D  TI';
JEDEC_TYPE='SOIC14';
CLASS='IC';
TECH='74LS';
end_body;
end_primitive;
END.
```

For more information on the `chips.prt` file, see the *Concept HDL Libraries Reference*.

Creating a Part Table File

The part table file associates a logical part with physical parts having varying physical properties. Each row in a part table corresponds to a physical part.

You can create a part table file (`.ptf`) using any text editor.

Given below is a sample part table file:

```
FILE_TYPE = MULTI_PHYS_TABLE;
PART ‘74F08’
CLASS = IC
:PACK_TYPE(OPT=’SOIC’)  ,PKG(OPT=’SOIC’) = JEDEC_TYPE, PART_NUMBER, COST, STATUS;
SOIC                  , SOIC   (1)      = SOIC14    , CDN0000-48 , .83 , PREF
DIP                   , DIP    (2)      = DIP14_3   , CDN0001-48 , .47 , NONPREF
LCC                   , PLCC20 (3)      = PLCC20    , CDN0003-48 , .91 , NONPREF
END_PART

PART ‘74F138’
CLASS = IC
:PACK_TYPE(OPT=’SOIC’)  ,PKG(OPT=’SOIC’) = JEDEC_TYPE, PART_NUMBER, COST;
SOIC                  , SOIC   (1)      = SOIC16    , CDN0000-38 , .93
DIP                   , DIP    (2)      = DIP16_3   , CDN0001-38 , .77
LCC                   , PLCC20 (3)      = PLCC20    , CDN0003-38 , .9
END_PART

PART ‘74F244’
CLASS = IC
:PACK_TYPE(OPT=’SOIC’)  ,PKG(OPT=’SOIC’) = JEDEC_TYPE, PART_NUMBER, COST;
SOIC                  , SOIC   (1)      = SOIC20W   , CDN0000-45 , .93
DIP                   , DIP    (2)      = DIP20_6   , CDN0001-45 , .87
LCC                   , PLCC20 (3)      = PLCC20    , CDN0003-45 , .71
```

```
END_PART
END.
```

For more information on part table files, see the *Concept HDL Libraries Reference*.

Adding a Component

1. Choose *Component* > *Add*.

   The *Add Component* dialog box (Component Browser) appears. If you select the *Category* tab, you can select the components under each category.

2. Select a library.

   Concept HDL displays the components in the library you select.

3. Select a component. The component attaches to the cursor.

4. Click on the drawing to place the component.

You can continue placing components until you choose another menu item or select *Done* from the pop-up menu.

To add a component with physical information,

1. Choose *Component* > *Add* to select a component using the library view or the category view.

2. Click on *Physical*.

   The *Physical Part Filter* dialog box appears. The *Physical Part Filter* displays information from the Physical Part Table file (`.ptf` file). Each row in the *Physical Part Filter* dialog box corresponds to a physical component associated with the logical part you select.

   If no physical information is available for the part, Concept HDL displays the Add Part dialog box that lets you select the Pack Type from the `chips.prt` file. Concept HDL annotates the `PACK_TYPE` property on the schematic.

3. Select a part under *Part Names* in the *Physical Part Filter*.

4. In the *Filters* row, you can decrease the number of available part choices based on specific criteria.

   For example, if you want to view only the physical parts of the package type `dip`, type `dip`, or `d*` in the filter space above the pack_type column and press *Enter*.

Note: You can select multiple logical parts in the logical view of the Component Browser and view their physical parts. This enables you to place the part with the desired physical information. The multiple selection of logical components and viewing their physical information is most useful when you have selected logical components by category.

To select multiple logical parts randomly, press the Ctrl button and select the parts.

➤ Select a row in the Physical part filter and click on the drawing to place the part with physical information.

Concept HDL annotates the schematic with the physical information depending on the options you set in the *Property Options* dialog box.

Modifying Components

To modify a single component,

1. Choose *Component > Modify*.
2. Select a component whose physical properties you want to modify.

   The *Physical Part Filter* dialog box appears with the filter set to the current physical property values in the component.
3. Click *Reset Filters* to display all rows in the part table file.
4. Select the desired row of physical properties to attach to the component you want to modify.

You can continue selecting and modifying components until you choose another menu command or select *Done* from the pop-up menu.

You can modify the physical properties of all components in a group if they are the same logical components.

To modify a group of components,

1. Choose *Group > Components > Modify*.

   The *Physical Part Filter* dialog box appears.
2. Select a row in the *Physical Part Filter* dialog box.

   The physical properties of all the components in the group are replaced with the row you select in the *Physical Part Filter*.

Replacing a Component

1. Choose *Component > Replace*.

   The *Component Browser* appears.

2. Select a component from the *Library View* or the *Category View*. If you choose from the *Library View*, you must specify a library in the *Library* box.

3. Click on the component in the schematic to replace it.

   The component is replaced with version 1 of the component that you selected in the *Component Browser*.

If you are in the pre-select mode in Concept HDL, you can replace multiple components by doing the following:

1. Use *Ctrl+click* or *SHIFT+click* to select multiple components.

2. Choose *Component > Replace* to display the *Component Browser*.

3. Select the component that should replace all the components.

To replace a component along with its physical properties

1. Choose *Component > Replace*.

   The Component Browser appears.

2. Select a component from the *Library*. You must specify a library in the *Library* box.

3. Click on the *Physical* button to load the *Physical Part Filter* dialog box.

   The *Physical Part Filter* loads the PPT file for the selected component.

4. Select a suitable row of physical properties from the Physical Part Filter and click *Close*.

5. Click on an existing component in the schematic to replace it.

You can continue replacing components until you choose another menu item or right-click to choose *Done*.

To replace components in a group

1. Set the current group.

2. Choose *Component > Replace*.

   The *Replace Component* dialog box appears.

3. Select the component that should replace all components in the current group.

   If you want to replace the components in the group with a component along with its physical properties, do the following:

   a. Click *Physical*.

      The *Physical Part Filter* dialog box appears.

   b. Select the appropriate row of physical properties from the *Physical Part Filter* dialog box and click *Close*.

All the components in the current group are replaced with version 1 of the component that you selected in the *Replace Component* dialog box.

Defining Physical Property Options

When you place a part with physical information in Concept HDL, you can specify physical property options to define the format and visibility of the properties.

For example, you can define the physical property options in such a way that properties such as `PART_NUMBER` and `TOL` will not appear on the schematic.

The physical property settings and definitions can be stored in a text file named `ppt_optionset.dat` and reused by a group for a given project. Concept HDL saves the settings for a project when you click *Save Options* in the *Property Options* dialog box. The *Property Option*s dialog box is accessed by clicking the *Options* button on the *Physical Part Filter* dialog box. The `ppt_optionset.dat` file is located in the project directory.

To define Physical Property Options,

1. Click *Options* in the *Physical Part Filter* dialog box.

   The *Property Options* dialog box appears.

2. In the *Property Order* box, you can modify the order in which the physical properties are displayed in the *Physical Part Filter* dialog box and annotated on the schematic.

   For example, if you move the `PACK_TYPE` property to the lowest level in the list, the Physical Part Filter displays it as the last column. When you click on a row to place a part with physical information, Concept HDL displays PACK_TYPE at the bottom of the list of properties.

3. Use the filter to filter physical property values based on the string you enter. For example, if you want the Physical Part Filter to display only that row of the PPT which has the value

of VOLTAGE as 63V, select the VOLTAGE property in the Property Order list box, enter 63 in the filter, and click *Apply*.

4. Specify how you want the physical property to be annotated on the schematic. If you select No, no physical properties are added on the schematic. If you select Name, only the names of the physical properties are displayed. If you select Value, only the values of physical properties are displayed. If you select Invisible, the physical properties are added on the schematic and are read by all tools, but they are not visible on the schematic.

5. Select the *Numeric Sort* check box to sort the rows using numeric sort. Numeric sort treats property values as numbers and sorts them accordingly (50<150).

   There are two types of sorting that Concept HDL employs to list the rows in a PPT file in the *Physical Part Filter* dialog box.

   - Numeric Sort

     In numeric sorting the lesser numbers are placed first followed by the greater numbers (50<150).

   - String Sort

     In string sorting, the first characters of the two values are compared. If they are the same, then the second characters are compared and so on. For example, if you compare 150 and 50, the first characters of 150 and 50 are compared. Since 1 < 5, 150 < 50.

6. Select the *Hide Column* check-box to hide the selected property column in the *Physical Part Filter* dialog box.

7. Click *OK* or *Apply* to apply the options you define.

8. Click *Save Options* to save all options to the `pptoptionset.dat` file. You can load these options in later sessions using the *Load Options* button. You can specify this file as the default option set file for Concept HDL by entering the path to the `ppt_optionset.dat` file in *Tools* > *Options* > *Paths*.

9. Click the *Load Options* button to load the option sets and the options from a `pptoptionset.dat` file.

Sample ppt_optionset.dat File

The syntax of the `ppt_optionset.dat` file is as follows:

```
( "VERSION 3.0"
   (
      ( OPTION_SET_ATTRIBUTES )
```

```
      ( PROPERTY_ATTRIBUTES )
   )
)
```

OPTION_SET_ATTRIBUTES has the following four fields:

Field	Description
optionsetname	A character string to identify the option set
totalprops	Total number of properties in the option set
Keyprops	Number of key properties in the set
Injprops	Number of injected properties in the set

PROPERTY_ATTRIBUTES has the following nine fields:

Field	Description
Proptype	Type of Property. 1 for key property and 0 for the injected property
Propname	Name of the property
Filterpattern	Filter pattern to be used for the property
SortType	1 to use numeric sort, 0 if numeric sort is not to be used
SortOrder	Order of properties in the optionset. 0 is used for the first order.
Annotate type	0 for normal, 1 for always, 2 for never
Annotate	0 for no annotation, 1 for name, 2 for value, 3 for both, and 4 for invisible
Visible	1 for property to be visible in the *Physical Part Filter* dialog box 0 for property to be hidden in the *Physical Part Filter* dialog box
Propcol	Column number of the property. 0 for the property in the first column

Given below is a sample ppt_optionset.dat file with the following option sets:

```
74LS00
74LS04

( "VERSION 3.0"

(
    ("74LS00" 12 6 6)
    (0 "JEDEC_TYPE" "*" 0 9 0 0 1 10)
    (0 "TOL" "*" 0 8 2 0 1 9)
    (1 "TYPE" "*" 0 3 1 2 1 4)
    (1 "PACK_TYPE" "*" 0 11 1 3 1 0)
```

```
    (1 "TOL" "*" 0 2 1 2 1 3)
    (0 "VALUE" "*" 0 6 2 0 1 7)
    (1 "VALUE" "*" 0 0 1 2 1 1)
    (1 "VOLTAGE" "*" 0 1 1 2 1 2)
    (0 "VOLTAGE" "*" 0 7 2 0 1 8)
    (0 "PART_NUMBER" "*" 0 5 0 0 1 6)
    (0 "ALT_SYMBOLS" "*" 0 10 0 0 1 11)
    (1 "LEADS" "*" 0 4 1 2 1 5)
)
(
    ("74LS04" 12 6 6)
    (0 "JEDEC_TYPE" "*" 0 10 0 0 1 10)
    (0 "TOL" "*" 0 9 2 0 1 9)
    (1 "TYPE" "*" 0 4 1 2 1 4)
    (1 "PACK_TYPE" "*" 0 0 1 2 1 0)
    (1 "TOL" "*" 0 3 1 2 1 3)
    (0 "VALUE" "*" 0 7 2 0 1 7)
    (1 "VALUE" "*" 0 1 1 2 1 1)
    (1 "VOLTAGE" "*" 0 2 1 3 1 2)
    (0 "VOLTAGE" "*" 0 8 2 0 0 8)
    (0 "PART_NUMBER" "*" 0 6 0 3 0 6)
    (0 "ALT_SYMBOLS" "*" 0 11 0 0 1 11)
    (1 "LEADS" "*" 0 5 1 2 1 5)
)
)
```

Breaking Up a Component

1. Choose *Component* > *Smash*.
2. Click a component in your drawing.
3. Select the discrete pieces that made up the component.

Changing Pin States on a Component

1. Choose *Component* > *Bubble Pins*.
2. Click a pin.

   Note: If the pins are part of a bubble group, you can choose *Bubble Pins* to convert the component from one form to another.

Example of Converting a Component from One Form to Another

For example, a `NOT` body is defined with both the `BUBBLED` and `BUBBLE_GROUP` properties attached:

```
BUBBLED=(B)
BUBBLE_GROUP=(A | B)
```

Because BUBBLED=(B), pin B is bubbled when the component is initially added to a drawing. If you choose *Bubble Pins* in the *Component* menu and click either pin A or B, the attached BUBBLE_GROUP property specifies that pin A is now the bubbled pin and pin B the un-bubbled pin.

Choosing a Version of a Component

1. Choose *Component* > *Version*.
2. Click a component in your drawing to display the next version.
3. Continue clicking on the component to view all the versions until the original version is displayed again.

Note: You can also run the `version` command using this stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes on page 112.

Mirroring Components or Blocks

1. Choose *Edit* > *Mirror*.
2. Click a component or a block.

Changing the Orientation of Components or Text

To rotate a component when adding it to the schematic:

1. Choose *Component* > *Add*.
2. Select a component to add.
3. Right-click and choose *Rotate* from the pop-up menu.
4. Choose *Rotate* from the pop-up menu continuously to rotate the component another 90 degrees each time.

To rotate a component that has already been placed in the schematic:

1. Choose *Edit* > *Rotate*.

2. Click a component.
3. Continuously clicking on the component rotates it another 90 degrees.

   Note: When you rotate a component, the associated property text appears either vertically along the left side of the component or horizontally above the component.

To spin a component:

1. Choose *Edit > Spin*.
2. Click a component.
3. Continuously click on the component to spin it again.

   Note: When you spin a component, the associated property text spins around with the component.

Sectioning a Component

1. Enlarge the drawing so that the component you want to section is clearly visible.
2. Choose *Component > Section*.
3. Click a component.

Each time you click, you select a different section of the physical component, and different pin numbers are displayed. The section assignment is removed each time you cycle through all the available sections.

When you section a component, the following properties are added on the component:

- SEC

  Assigns a logical component to a particular section within a physical part.

- SEC_TYPE

  Identifies the package type in the *chips.prt* file used to get pin number assignments when sectioning a part.

If you section a component in the Occurrence Edit mode in Concept HDL, the text <<ERROR>> might be displayed against a pin on the component. If you package a design that has a component with this error, packaging fails with the following error:

```
ERROR(1134): PN <n> does not belong to SEC <n>.
```

To correct this problem, swap the pins in the Occurrence Edit mode or section the component in the Hierarchy or Expanded mode in Concept HDL.

Swapping Pins on a Component

A component must be sectioned before you can swap pins on it.

1. Choose *Component > Swap Pins*.
2. Click the two pins you want to swap.

   Note: Properties attached as a result of swapping pins can only be deleted or moved, not changed. You should not change the PN property. After swapping, $PN becomes the hard property PN.

Ways to Determine if a Component Has Bus-Through Pins

- Choose *Display > Pins* to display an asterisk at the location of every pin.
- Choose *Display > Pin Names* to display the pin names for the component.
- Bus-through pins have the same name as the corresponding visible pin.
- Look at the symbol view of the component to see if the component is defined with a bus-through pin.

Deleting a Library Component (Cells, Views, and Files)

1. Choose *File > Remove*.
2. In the scroll area of the *View > Remove* dialog box that appears:
   a. Select a cell to delete the entire cell.
   b. Click + next to the cell name to expand the hierarchy, and select a view to delete.
   c. Click + again to expand the hierarchy, and select a page to delete.
3. Click *Remove*.

Caution

You must not delete cells, views or files from Windows Explorer or the UNIX or DOS command prompt. This can create problems in the design.

Creating a SYNONYM

To create a SYNONYM for a signal:

1. In Concept HDL, choose *Component > Add*.
2. In the *Library* box in the Component Browser, select *Standard*.
3. Select SYNONYM from the cell list of Component Browser.

   The synonym symbol gets attached to the cursor.
4. Click in the Concept HDL drawing area to place the symbol.
5. Attach the signal for which you want to create a synonym to the left pin of the SYNONYM symbol.
6. Attach the SYNONYM name to the right pin.

You must follow the following rules for using the SYNONYM symbols:

- Connect signals of the same assertion: both must be high or both must be low.
- Connect signals of the same width.
- Synonym symbols cannot be used if you want to generate VHDL text for the schematic. Use ALIAS symbols instead.

Tapping a bit with a TAP symbol

1. Attach a TAP symbol to the bus.
2. Use the *Text > Change* command to set the value of the BN property to the actual bit number that you want to tap.

Creating a Page Border Symbol

The first step while creating any design is to add a page border. You can have a design without page borders, but it is a good design practice to add page borders. Page borders provide a convenient way of documenting information such as the date, the design name, the page number, the engineer's name, the company logo and so on, on the schematic. Concept HDL allows you to specify the default page border that you want to be used automatically every time you create a schematic page. For more information, see Setting Automatic Page Borders on page 90.

Page borders are required when you cross reference a design. When you plot a schematic, it is often difficult to trace the location of a signal or instances of a part. CRefer traces the signals and parts in a schematic and annotates the location of each one in text reports.

CRefer writes the page number and the location of the part or signal in relation to the page border. For more information, see Chapter 17, “Cross-Referencing Your Design.”

The Cadence Standard library provides six standard page borders—`A SIZE PAGE` to `F SIZE PAGE`—that you can use in your design. For more information, see PAGE Borders on page 624.

This section describes the procedures for customizing a page border in the Standard library or creating a page border of your own.

- Customizing a Page Border in the Standard Library on page 199
- Creating a Page Border of Your Own on page 200

Customizing a Page Border in the Standard Library

Cadence recommends that you customize a page border in the Standard library instead of creating a page border of your own. This is because page borders are created by drawing wires and adding notes, and it is time consuming to create a page border of your own.

To customize a page border in the Standard library, do the following:

1. Create a new project using Project Manager.
2. Choose *Tools > Concept* in Project Manager to start Concept HDL.
3. In Concept HDL, choose *File > Open*.

   The *View Open* dialog box appears.
4. Select `Standard` library in the *Library* drop-down list.

   The list of components in the library are displayed.
5. Select the page border that you want to customize.

   The page border name is displayed in the *Cell* field.
6. Select *Symbol* from the *View* drop-down.
7. Click *Open* to open the symbol for the page border in Concept HDL.
8. Choose *File > Save As*.

   The *View Save As* dialog box appears.
9. From the *Library* drop-down list, select the library in which you want to save the page border.

10. Specify the name of the page border in the *Cell* field.

11. Click *Save*.

12. Make the necessary changes in the page border. For example, you can do the following:

- Choose *Wire* > *Draw* to add boxes for placing notes, or add your company logo by drawing wires. For example, the Cadence logo in the `CADENCE A SIZE PAGE` page border symbol in the Standard library was created by drawing wires.
- Choose *Text* > *Note* to add notes, copyright information, non-disclosure information and so on.
- Add custom text. For more information, see Adding Custom Text on Page Borders on page 201.

13. Choose *File* > *Save* to save the changes.

Maintain the page border symbol in a reference library so that other users can use the page border.

You can now use the page border symbol on your schematic pages. If you want to cross reference a design that uses the page border, define the page border in the `cref.dat` file located at `<your_install_dir>/share/cdssetup/creferhdl/`. For more information on the `cref.dat` file, see Cref Data File on page 484.

Creating a Page Border of Your Own

1. Create a new project using Project Manager.

2. Choose *Tools* > *Concept* in Project Manager to start Concept HDL.

3. Run the following command in the Concept HDL console window:

```
edit <page_border_name>.sym.1.1
```

Concept HDL creates a symbol drawing named `<page_border_name>.sym.1.1` and places the ORIGIN symbol from the Standard library on the drawing. The ORIGIN symbol is placed at coordinates `(0,0)` on the drawing.

Note: You should not move the ORIGIN symbol from this location on the drawing.

Tip

You can choose *Display* > *Coordinate* and click on the ORIGIN symbol to display the coordinates in the Concept HDL console window

4. Choose *Text* > *Attributes* and click on the ORIGIN symbol.

The *Attributes* dialog box appears.

5. Click *Add*.
6. Type `COMMENT_BODY` in the *Name* field.
7. Type `TRUE` in the *Value* field.
8. Click *OK* to close the *Attributes* dialog box.
9. Choose *Wire > Draw* to draw the page border.
10. Choose *Text > Note* to add zones on the page border. The zones are used by CRefer to display the location of schematic objects in the CRefer text reports. For more information, see Creating Zones on Page Borders on page 203.
11. Choose *Wire > Draw* to add boxes for placing notes or add your company logo by drawing wires. For example, the Cadence logo in the `CADENCE A SIZE PAGE` page border symbol in the Standard library was created by drawing wires.
12. Choose *Text > Note* to add notes, copyright information, non-disclosure information and so on.
13. Add custom text. For more information, see Adding Custom Text on Page Borders on page 201.
14. Choose *File > Save* to save the changes.

    Maintain the page border symbol in a reference library so that other users can use the page border.

You can now use the page border symbol on your schematic pages. If you want to cross reference a design that uses the page border, define the page border in the `cref.dat` file located at `<your_install_dir>/share/cdssetup/creferhdl/`. For more information on the `cref.dat` file, see Cref Data File on page 484.

Adding Custom Text on Page Borders

You can add custom text in page borders to display page numbers, design information, cross referencing information and so on, on the schematic pages. For more information, see Working with Custom Text on page 239.

Note: You must attach the custom text to the ORIGIN symbol on the page border.

The page border symbol displays the format string for the custom text. When the page border is instantiated on a schematic page, the values of custom variables are substituted. For example, add the following custom text on the page border symbol:

`Page <CON_PAGE_NUM>`

When the page border is instantiated on a schematic page, the custom variable `CON_PAGE_NUM` will take its actual value on each page. For example, `Page 1` or `Page 2`.

See Adding Custom Text on page 245 for the procedure for adding custom text.

Tip

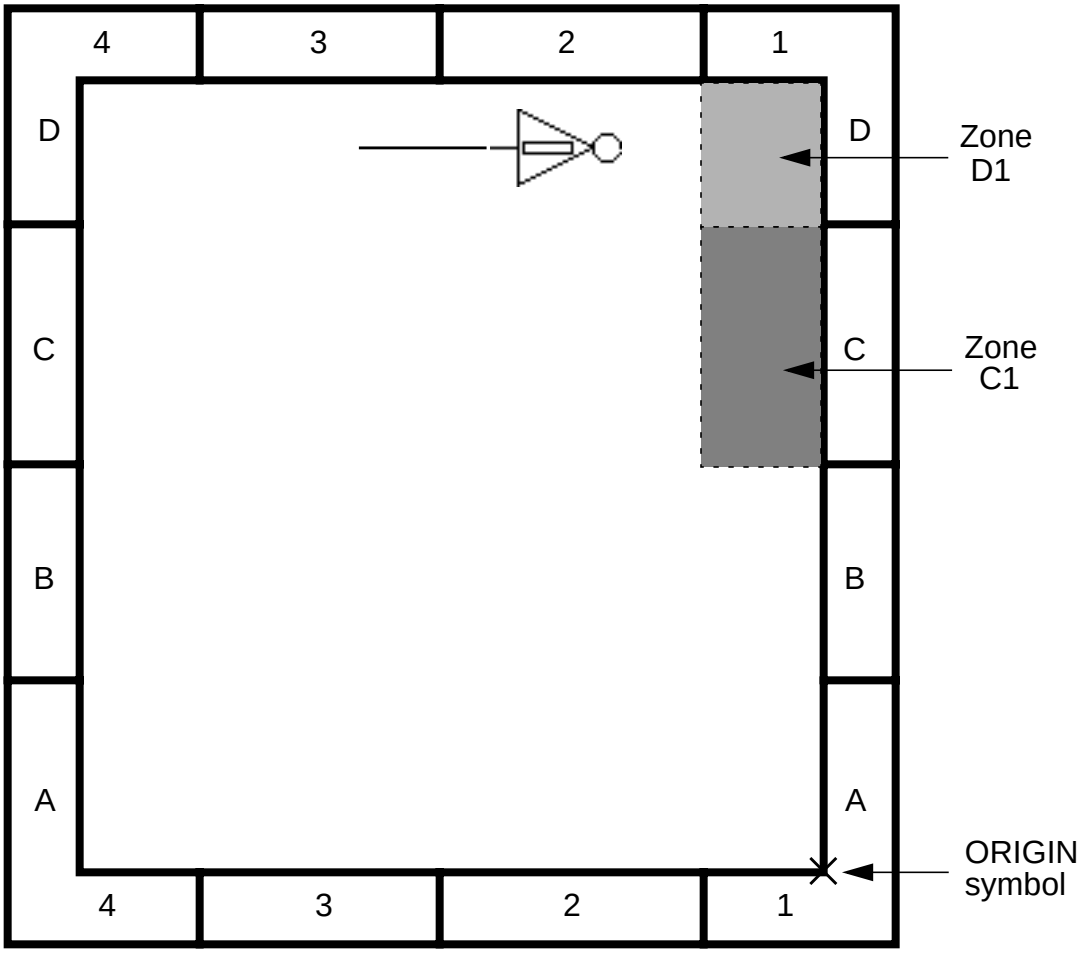
To quickly locate the origin of a page border symbol, run the following console window commands:

```
find origin
next
```

The origin of the page border symbol is highlighted.

Creating Zones on Page Borders

You can create zones on page borders as shown in the following figure.



The zones are used by CRefer to display the location of schematic objects in the CRefer reports. For example, the Crefparts report will display the location of the `ls04` component in the above schematic as:

```
<value of LOCATION property> 74LS04 <cell_name> [ <page_number>D2 ]
```

For example, if the `ls04` component that has the `$LOCATION=U1` property is located in zone `D2` on a schematic page `ANALOG_IO.SCH.1.8`, the Crefparts report will display the location of the `ls04` component as:

```
U1 74LS04 ANALOG_IO [ 8D2 ]
```

9

Working with Wires

This section describes the procedures for working with wires in Concept HDL.

About Signals and Connectivity

It is important to identify each of the primary inputs and outputs of the circuit and other important signals with a name. Signal names identify signals on the drawing. Signals with the same name are interpreted as the same signal. This is how Concept HDL connects signals across multiple pages of a drawing.

Signal names also let you enter additional information:

Assertion level	Describes the active state of the signal when asserted. By convention, a signal is active high for positive logic and active low for negative logic. An asterisk * and ‘-‘ represents active low. Two signals with the same name but different assertion levels are not the same signal. For example, Concept HDL treats A* and – A as low-asserted pins.
Signal bits	Signals can have a single bit or multiple bits. The bit portion of the signal name is called the bit subscript and is always enclosed in angle brackets, like this: <3..0>. A signal without a signal bit is called a scalar. A signal with a bit subscript can be a scalar or a vectored signal.
Properties	Describe characteristics of the signal, control how the compiler interprets the signal, or conveys physical information.

Concept HDL handles signal names as properties. For example, attaching a signal called BUS ENABLE to a wire is equivalent to attaching a property SIG_NAME=BUS ENABLE to that wire. In the symbol, the SIG_NAME properties are understood as PIN_NAME properties and can only be attached to pin connections.

The names you attach to the signals in the drawing are written into the connectivity file that Concept HDL creates when you save the drawing.

About Bus Taps

Concept HDL provides several different bus taps for use in schematics. These bus taps are in the Standard Library.

The most convenient way to tap buses is to choose *Wire > Bus Tap*. You can choose *Tools > Options* and specify the tap to use in the Graphics tab.

You can use other tap symbols (tap.body, bustap.body, msbtap.body, and lsbtap.body) or create your own tap symbol. For guidelines for creating tap symbols, see Guidelines for Creating Tap Symbols on page 617.

When you add a tap using *Wire > Bus Tap*, the BN property is added to the bus tap. Concept HDL understands that if you have a bus named <20..5> and you attach a tap to it with the BN property set to 7, then you are tapping bit 7, not bit 12.

About Bus Names

Concept HDL supports several bit numbering syntax conventions. Because the signal name syntax affects library parts and many design tools, a single site must use the same syntax system wide. Bit subscripts can use two dots (..) or a colon. Bit ordering can be most significant bit to least significant bit (msb to lsb) or vice versa.

Bus Name	Associated Signal Name
A<3..0>	A<3>, A<2>, A<1>, A<0>
A<0..3>	A<0>, A<1>, A<2>, A<3>
A<0>	A<0>
A<7..0:2>	A<7>, A<5>, A<3>, A<1>

Drawing a Wire Manually

To draw a wire without naming it

1. Choose *Wire > Draw*.

2. Click a pin on a component.
3. To change the orientation of the wire as you draw it, click right and choose *Orientation* from the pop-up menu.
4. Each time you choose *Orientation* you can change the bend of the wire.
5. Click again wherever you want the wire to bend, or click a pin on another component.

To name a wire when you draw it

1. Choose *Wire > Draw*.
2. Click right and choose *Signal Name…* from the pop-up menu.
3. Type a signal name in the *Signal Name* box.
4. Click *OK*.
5. Click wherever you want the wire to bend, or click a pin on another component.

Tips for Drawing Wires

To...	Do this...
End a wire at a pin, dot, or other wire	Click left
Snap a wire to the nearest pin	Click Ctrl+right
Change the bend of the wire	Click Ctrl+left or click right and choose *Orientation* from pop-up menu
End a wire in a free space	Click twice at the final point

Auto-Routing a Wire

1. Choose *Wire > Route*.
2. Click the edge of a component, then click the edge of another component.

Stretching a Wire

1. Choose *Edit > Move*.
2. Click a wire end and stretch the wire to the desired length.

Bending a Wire

1. Choose *Wire > Draw*.
2. Begin drawing a wire.
3. Click right and choose *Orientation* from the pop-up menu.
4. The bend of the wire changes from orthogonal to diagonal. You can continue to cycle through different wire bends by choosing *Orientation* from the pop-up menu
   - ❑ Each time
   - ❑ Once, then press Ctrl+Left

Splitting a Wire

1. Choose *Edit > Split*.
2. Click on a wire and move the cursor down or up.

   Concept HDL displays the wire you are working with as red.
3. Double-click.

   Concept HDL displays one of the wire ends as red, indicating you can work with it separately.

Snapping a Wire to the Nearest Pin

1. Choose *Wire > Draw*.
2. Press *Ctrl* + click right.

   Concept HDL draws a wire starting at the closest pin.
3. Press *Ctrl* + click right to snap the other end of the wire to the nearest pin.

Naming a Signal

To name an existing wire

1. Choose *Wire > Signal Name*.

   The *Signal Name* dialog box appears.

2. Type one or more signal names on separate lines.
3. Select the wire(s) you are naming in the same order you entered them in the *Signal Name* dialog box.

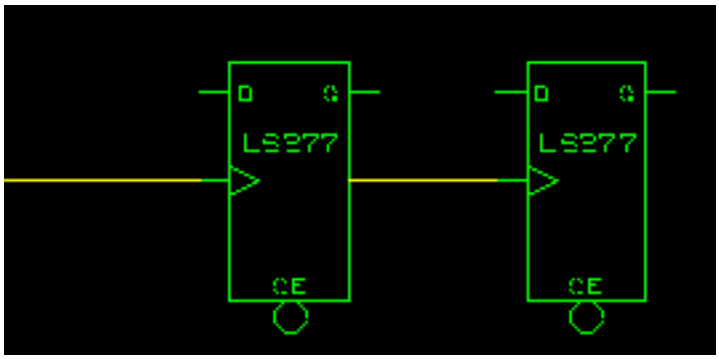
To name a wire when you draw it

1. Choose *Wire > Draw*.
2. Click right and choose *Signal Name…* from the pop-up menu.
3. Type a signal name in the *Signal Name* box.
4. Click wherever you want the wire to bend, or click a pin on another component.

Wiring Bus-Through Pins

1. Locate bus-through pins.
2. Choose *Wire > Draw*.
3. Click a component at the location across from the input pin, and connect the wire to an input pin on another component.

Example



Marking Wire Connections

1. Choose *Wire > Dot*.
2. Click a wire intersection.

Naming Signals on a Bus

1. Choose *Wire > Bus Name*.
2. Type a name in the *Bus Name* box.
3. Specify *MSB* (most significant bit), *LSB* (least significant bit), and *Increment*.
4. Click above the first wire.

   Concept HDL attaches a flexible line to the cursor.
5. Move the cursor so that the line crosses all the taps and click again.

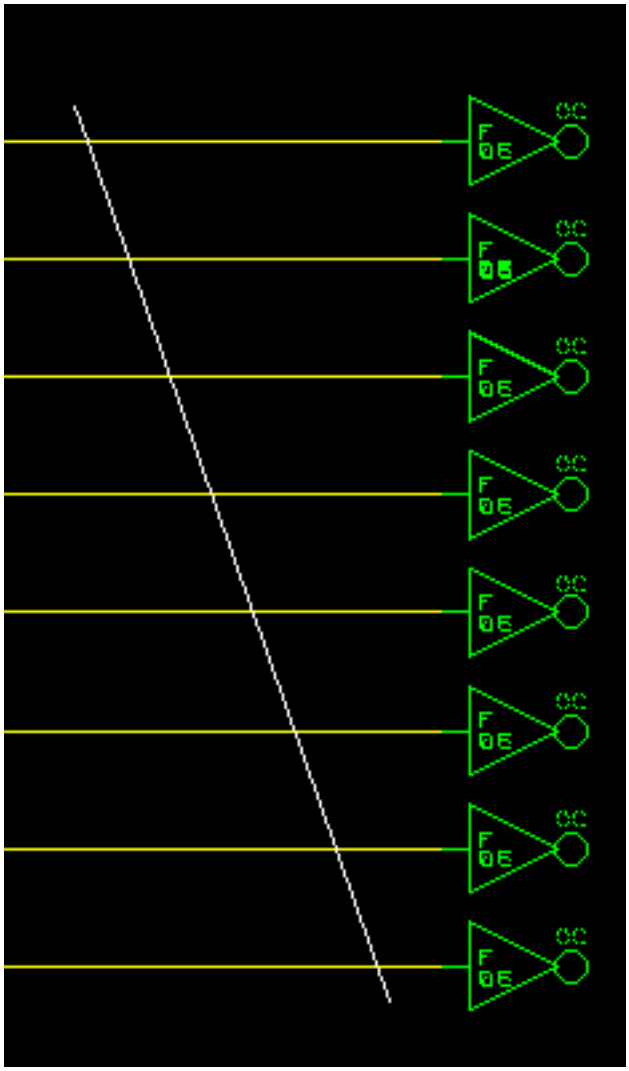
   The MSB value is placed on the tap closest to the first location you click, and the LSB value is placed on the tap closest to the second location you click.

Example

Say you want to name a 7-bit bus. You might specify:

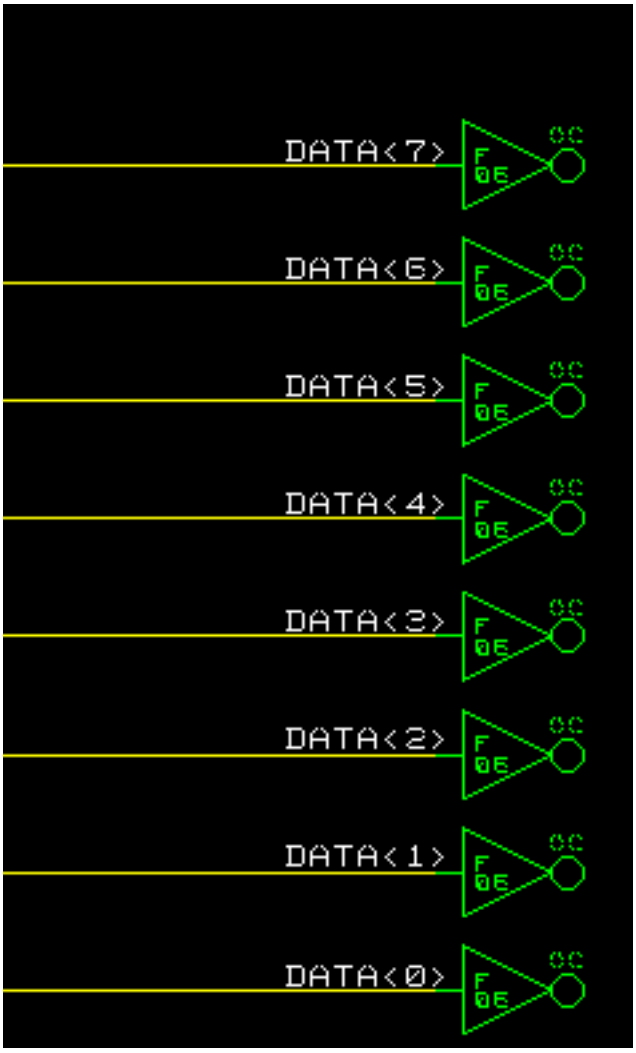
- data as the bus name
- 7 as the most significant bit
- 0 as the least significant bit
- 1 as the increment

In this example, the first click is above the top bit.



Before

The second click is below the bottom bit. Concept HDL draws a line between the two points. <7> is placed on the tap closest to the first location you click and <1> on the tap closest to the second location you click.



After

Then the bus names and values appear.

Specifying a Tap Symbol

1. Choose *Tools* > *Options*.

2. In the *Symbols* box on the *Graphics* tab, type the name of the tap symbol you want to use in the *Tap Symbol* box.

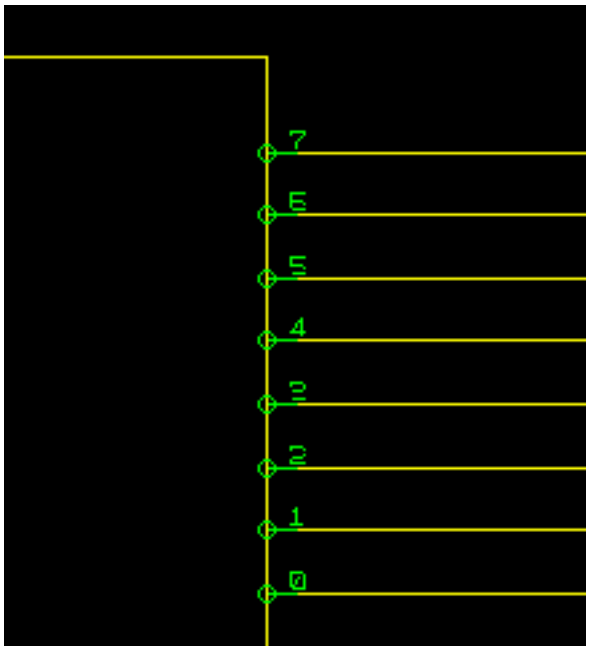
Attaching Values to Bus Taps

1. Choose *Wire > Bus Tap Values*.
2. Specify *MSB* (most significant bit), *LSB* (least significant bit), and *Increment*.
3. Click above the first wire.

   Concept HDL attaches a flexible line to the cursor.
4. Move the cursor so that the line crosses all the taps and click again.

   The MSB value is placed on the tap closest to the first location you click, and the LSB value is placed on the tap closest to the second location you click.

+



Changing Wire Thickness and Pattern

Choose one of the following in the *Wire* menu.

Thick	Makes the wire thick to indicate a bus.
Thin	Is the normal thickness for a wire.
Pattern…	Lets you choose from a variety of wire patterns.

10

Working with Properties and Text

This section describes the procedures for working with properties and text in Concept HDL.

About Properties

Properties (also called attributes) are used to convey information about a design. There are four types of Concept HDL properties.

- System properties - these are system-assigned properties that are attached to wires (nets) or pins.
- Schematic properties - these are user-assigned properties that can be attached to components, wires (nets), or pins.
- Symbol properties - these are librarian-assigned properties that are attached to a part through the part symbol drawing, the `chips.prt` file, or the part table file (PPT).
- Occurrence properties - these are properties assigned to schematic objects to define different values for the same property.

Definitions of the Concept HDL properties include these classifications. In addition, you can use extensions to Concept HDL properties.

Properties consist of a name and value. Using the *Text > Property Display* menu command, you specify whether Concept HDL displays the property name alone, the property value alone, both, or neither. A property name can combine letters, numbers, and underscores, but the first character must be alphabetic. A property value can include space and punctuation marks.

The following types of property value entries are supported:

25oct98 10:31:46.03

(size + 4) / 5 + 35 MOD A

This is a long property value

@#$%*( ) { } [ ] < >

Note: The maximum permissible length for a property name is 31 characters and that for a property value is 255 characters.

While Concept HDL does not interpret most properties (it passes them to other system tools), it does interpret these properties:

- LAST_MODIFIED
- PIN_NAME
- SIG_NAME
- Properties added by *Tools > Back Annotate*, *Component > Section*, and *Component > Swap Pins*
- PATH

Adding Properties

There are several ways to add properties to a drawing. A good way to add several properties to an object and be sure that their names and values are correct is to use an attribute file as a template.

Copying Properties

You can copy most properties on the drawing. Default properties and properties that you add are automatically included in copies made of components. If you change a default property on a component, the property on a copy of the component also changes.

Properties you cannot copy are:

- pin properties
- wire properties
- unnamed signals
- PATH properties

Moving Properties

Concept HDL moves properties with the object to which they are attached, or you can move properties independently. You cannot move the PATH property between drawings.

Adding Properties

To add one property at a time

1. Choose *Text* > *Property*.
2. In the *Property* dialog box, enter a name in the *Property Name* box and a value in the *Property Value* box.
3. Click *OK*.
4. Click the object to which you are attaching the property.
5. Click near the object to indicate where to display the property information.

   As the default, Concept HDL displays only the property value. Choose *Text* > *Property Display* to modify how properties are displayed.

Note: You can also run the `property` command using this stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes on page 112.

You can also add a property using the property command.

To add many properties together using the Attributes dialog box

1. Choose *Text* > *Attributes* or type `attribute` in the console window.
2. Select an object.

   The object you select is highlighted, and the *Attributes* dialog box displays the properties for the object.
3. Click *Add* in the *Attributes* dialog box.

   An empty row appears.
4. Enter a property name in the *Name* box and a property value in the *Value* box.
5. Adjust property visibility and alignment as needed.
6. Click *OK* to save the changes and close the dialog box.

To add occurrence properties

1. Choose *Tools > Expand Design*.
2. Choose *Tools > Occurrence Edit* or type attribute in the console window.
3. Choose *Text > Attributes*.
4. Select an object.

   The object you select is highlighted, and the *Attributes* dialog box displays the properties for the object.
5. In the *Attributes* dialog box, click *Add*.

   An empty row appears.
6. Enter a property name in the *Name* box and a property value in the *Occurrence Value* box.
7. You can also replace the occurrence value for an existing property. However, you cannot change or delete schematic values that are shaded gray.
8. Adjust property visibility and alignment as needed.
9. Click *OK*.
10. Click near the object to indicate where to display the property information.

    As the default, Concept HDL displays only the property value. Choose *Text > Property Display* to modify how properties are displayed.

Displaying and Modifying Property Attributes

To display property attributes

1. Choose *Text > Attributes*.
2. Select an object.

   The object you select is highlighted, and the *Attributes* dialog box displays attributes for the object.

Note: You can also run the `attribute` command using this stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes on page 112.

To display net properties from net synonyms

1. Choose *Tools > Expand Design*.
2. Choose *Text > Attributes*.
3. Select the net.

   If you need to find the net in your design, use *Tools > Global Navigate* to find a net that spans the design hierarchy or multiple pages in a design.

   The net is highlighted. The *Attributes* dialog box displays the object name and a list of properties and property values, as well as the canonical name of the net synonym for which the properties were defined.

To modify a property

1. Click a property in the *Attributes* dialog box.
2. Highlight a property name or value and type a new name or value, or adjust property visibility and alignment as needed.
3. Click *OK* for modifications to appear on the drawing, or click *Cancel*.

To delete a property

1. Click a property in the *Attributes* dialog box.
2. Click *Delete*.

   The property is deleted from the *Attributes* dialog box.
3. Click *OK* to delete.

Making an Attribute File

1. Display the *Attribute* dialog box.
2. In the *Attribute* dialog box, choose *File > Load Attributes*.
3. Select a file in the directory browser that appears and click *Open*.

Properties already on the object are listed in the *Attributes* dialog box first, followed by properties from the attribute file you loaded.

4. Delete unnecessary properties from the *Attributes* dialog box.
5. Change the values of properties and display options if necessary.
6. In the *Attribute* dialog box, choose *File > Save Attributes*.
7. Specify a new filename in the *File* box of the *Save As* window that appears.

Important

Click *Cancel* to close the *Attribute* dialog box if you do not want to apply these properties to the object you selected to open the dialog box.

Adding Text

To add individual text lines

1. Choose *Text > Note*.

   The *Note* dialog box is displayed.

2. Type text in the *Notes* section of the dialog box.

   You can enter several strings before placing them on the drawing.

3. Enable *Queue* to place text lines in the order you enter them. Enable *Select* to choose a text line and place it (the text you select for placement is highlighted in the *Note* dialog box).

Note: You can also run the *note* command using this stroke pattern:

For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes on page 112.

To add text from a file

1. Choose *Text > File*.

   The *Open* dialog box appears.

2. Navigate to the text file you want and highlight the file name.
3. Click *Open*.
4. Click in a blank space in the drawing.

Note: To include a blank line, type a space on the line in the file before adding it to the drawing. The line is otherwise ignored when the file is added to the drawing. Tabs and nonprintable characters in the file are displayed as a # sign.

Modifying Text

To modify single text items

1. Choose *Text > Modify*.
2. Select the text to be modified.

   Note: You can select multiple text items.
3. Use the arrow keys to position the cursor in the text line.
4. Press any character key to add text, press *Del* or *Backspace* to delete a character, press *Ctrl+K* to delete text from the cursor to the end of the line, press *Enter* or *Return* to move to the next text selection, or press *Esc* to end changes. Alternatively, you can use the pop-up menu to display the text change editor.
5. Choose Done from the pop-up menu when you have finished making modifications.

   Note: You can also run the `change` command using this stroke pattern:

   For more information on strokes and a list of available stroke patterns, see Running Commands with Strokes on page 112.

To modify text on grouped objects

1. Choose *Group > Set Current Group* to specify a group.

   Note: You can highlight the current group to emphasize which group you are working with (*Group > Highlight [x]*).
2. Choose *Group > Text Change [x]*.

   Note: A message might display with a reminder that you cannot change section

properties.

3. Right-click and choose *Editor* from the pop-up menu.
4. Edit text in the text change editor (*Ctrl+E*) and save (*File > Save* or *File > Save As*) before exiting the editor (*File > Exit*).
5. Right-click and choose *Done* from the pop-up menu.

Resizing Text

To increase text size

1. Choose *Text > Increase*.
2. Click the text line whose size you want to increase.

To decrease text size

1. Choose *Text > Decrease*.
2. Click the text line whose size you want to reduce.

Setting the Text Size

To set the text size

1. Choose *Text > Set Size.*

   The *Text Set Size* dialog box appears.
2. Enter the text size in inches.
3. Click *OK*.
4. Click on a note, a property, or a group to change the text size.

   Concept HDL changes the text size to the size you have specified.

Note: Concept HDL accepts the text size between 0.009 inches and 1.74 inches.

Changing the Text Editor

1. Choose *Tools > Options*.

2. Select the *Text* tab.
3. In the *Text Change Editor* box, specify the editor you want to use.
4. Click *OK*.

Adding Port Names from the Corresponding Symbol

1. Choose *Text > Port Names*.
2. Click in the drawing.
3. Pin names from the symbol are listed.

Swapping Notes or Properties

1. Choose *Text > Swap*.
2. Click a note or property.
3. Click a second note or property.
4. Concept HDL swaps the text line in one location on the drawing with the text line in the other location.

Reattaching a Property from One Object to Another

1. Choose *Text > Reattach*.
2. Click a property to be reattached.
3. Concept HDL draws a line from the property to the current cursor position.
4. Click an object that will be the new attachment point for the property.
5. If required, choose *Edit > Move* to move the property closer to its new attachment point.

Specifying Property Display for Text

1. Choose *Text > Property Display > Name/Value/Both/Invisible*, depending on whether you want to
   - ❑ Display only the property *Name*
   - ❑ Display only the property *Value*

- ❑ Display both the property name and the value.
- ❑ Make properties *Invisible*

2. Select a property.

Specifying Visibility of Pin Properties

Concept HDL allows you to specify the default visibility of pin properties. When you add a component on the schematic, the visibility of the properties on the pins of the component is controlled by the visibility option you have selected.

To set the default visibility for pin properties, do the following:

1. Choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Text* tab.
3. Select the pin property visibility option.

Select	To
Invisible	Make all pin properties invisible when you add a component in your schematic. The pin property will be invisible in your schematic even if the visibility of the property on the pin on the symbol for the component is set to *Name*, *Value* or *Both*.
Defined By Component	Make all pin properties visible or not depending on how the property visibility is defined on the pin on the symbol for the component.

4. Clock *OK* to save the changes.

Specifying Visibility of Pin Numbers

Pin numbers are assigned to the pins of a component in a design when you package the design. After you package the design, pin numbers are displayed in the Occurrence Edit mode in Concept HDL. After you backannotate the design, pin numbers are displayed in the Hierarchy and Expanded mode in Concept HDL.

You may not want the pin numbers on some components to be displayed on the schematic because it clutters the schematic. You may also want the pin numbers on some components to be always visible by default because you want to plot the pin numbers on such components.

The value of the `$PN` property on a pin is the pin number for the pin. Concept HDL allows you to specify the default visibility of pin numbers on a component in the schematic by adding the `$PN=?` and `$PN=#` properties on the pins on the symbol for a component. For more information, see the following sections:

- Making Pin Numbers Visible by Default on page 225
- Making Pin Numbers Invisible by Default on page 225

Making Pin Numbers Visible by Default

To make the pin numbers on a component visible by default:

➤ Add the `$PN=?` placeholder property on the pins on the symbol for the component.

Before you add the component on the schematic, set the *Pin Property Visibility* option to *Invisible* in the *Text* tab of the *Concept Options* dialog box.

After you package the design, the pin numbers will be visible on the schematic in the Occurrence Edit mode. When you run *Tools > Back Annotate*, the pin numbers will be visible on the schematic in the Hierarchy and Expanded mode.

After you package the design, the `$PN=?` property becomes `$PN=<pin_number>`. If you now change the visibility of `$PN` property on the component to *None*, the pin number will not be visible anymore.

Making Pin Numbers Invisible by Default

To make the pin numbers on a component invisible by default:

➤ Add the `$PN=#` placeholder property on the pins on the symbol for the component.

Before you add the component on the schematic, set the *Pin Property Visibility* option to *Invisible* in the *Text* tab of the *Concept Options* dialog box.

After you package the design, the pin numbers will remain invisible on the schematic. The `$PN=#` property becomes `$PN=<pin_number>`. If you now change the visibility of `$PN` property to *Value*, the pin number will become visible on the schematic.

Case-Sensitivity in Property Names and Values

In the HDL environment, case sensitivity of property names and values is supported in the front-to-back flow. You can define properties in the schematic, occurrence property file, PPT and `chips.prt` files. They can also be fed back from the board.

Note: In Concept HDL all property names and values are automatically uppercased. To change this behavior,

1. From the Tools menu in Concept HDL, select *Options* > *Text*.
2. Deselect the *Upper-Case Input* check box.

   You can now enter lower or mixed case property names and property values. They will not be automatically uppercased by Concept HDL and will be displayed in the schematic in the same case that they are entered.

By default, for all such properties

- the name of the property is uppercased by the tools
- the case of the property value is preserved.

You can change this default behavior for specific properties by attaching specific attributes to them in the `cdsprop.paf` file, which is located at

<your_inst_dir>/share/cdssetup

The `cdsprop.paf` is a text file. You can edit this file to make necessary changes to the case sensitivity of property names and values. You can place this file in the project directory if the attributes have to be applied only to that project.

To indicate that the case of a property name should be preserved, use the keyword `preservename`. To indicate that the case of a property value should not be preserved, use the keyword `uppercasevalue`.

Example:

```
alt_symbols: uppercasevalue
```

```
\TimingVersion\:preservename
```

In the example, ALT_SYMBOLS is assigned the keyword uppercasevalue. If you now specify the property, its value will always be uppercased.

The TimingVersion property is specified within backslashes (\) since the `cdsprop.paf` file is in the VHDL namespace. (In VHDL, backslashes are used to denote that the properties are case-sensitive.

TimingVersion is assigned the keyword preservename. So the property name will not be uppercased to TIMINGVERSION.

Case Insensitive Property Values

There are certain properties whose values should always be uppercased because they do not support case-sensitive values. These properties are assigned the keyword uppercasevalue in the default cdsprop.paf located in the installation hierarchy *<your_inst_dir>*/share/cdssetup.

The properties defined in `cdsprop.paf` whose values should always be uppercased are:

- ALT_SYMBOLS
- BODY_NAME
- CDS_NAME_OF_PART
- CDS_LOCATION
- CDS_PN
- GROUND_NETS
- JEDEC_TYPE
- LOCATION
- $LOCATION
- MERGE_NC_PINS
- MERGE_POWER_PINS
- NC_PINS
- PACK_TYPE
- PACK_SHORT

- PART_NAME
- PIN_NUMBER
- POWER_GROUP
- POWER_PINS
- PN
- $PN
- SD_SUFFIX_SEPARATOR
- SUBDESIGN_SUFFIX

Note: If you remove these assignments from the `cdsprop.paf` file, Packager-XL generates a warning message and these property values will continue to be treated as case-insensitive.

You can get an updated list of properties from the `cdsprop.paf` file located in your installation hierarchy <*`your_inst_dir`*>/`share/cdssetup`.

Case Insensitive Property Names

There are some properties whose names are always treated as case-insensitive. These property names are always uppercased.

The following is a list of case-insensitive properties:

- ALT_SYMBOLS
- BIDIRECTIONAL
- BODY_NAME
- BODY_TYPE
- CDS_LONG_PART_NAME
- CDS_NAME_OF_PART
- CDS_PARENT_CHIPS_PHYS_PART
- CDS_PARENT_PPT
- CDS_PARENT_PPT_PART

- CDS_PARENT_PPT_PHYS_PART
- CDS_PHYS_NET_NAME
- CDS_PRIM_FILE
- CDS_SEC
- CDS_LOCATION
- CDS_PN
- GROUP
- HAS_FIXED_SIZE
- INPUT_LOAD
- JEDEC_TYPE
- LOCATION
- $LOCATION
- MERGE_NC_PINS
- MERGE_POWER_PINS
- NC_PINS
- NO_BACKANNOTATE
- OUTPUT_LOAD
- OUTPUT_TYPE
- PACK_TYPE
- PACK_IGNORE
- PACK_SHORT
- PART_NAME
- PATH
- PINCOUNT
- PIN_GROUP
- PIN_NUMBER
- POWER_GROUP

- POWER_PINS
- PN
- $PN
- PRIM_FILE
- ROOM
- SEC
- $SEC
- SD_SUFFIX_SEPARATOR
- SIZE
- SUBDESIGN_MASTER
- SUBDESIGN_SUFFIX
- WIRE_GATE
- XY

Note: If you assign any of these properties as case-sensitive (by assigning the keyword preservename) in the `cdsprop.paf` file, Packager-XL will generate a warning message and continue to treat these as case-insensitive.

Case Sensitivity and PPTs

In Concept HDL, when you select a part with physical information, Concept HDL compares the key property name on the Concept HDL canvas and the key property name in the physical part table file. This comparison is case-insensitive.

For example, if the key property name in the PPT header is abcd and the property name on the schematic instance is ‘ABCD’, the match will still take place.

For properties that have the keyword uppercasevalue in `cdsprop.paf`, the comparison for the value is always case-insensitive. For example, the PACK_TYPE property has the keyword uppercasevalue in `cdsprop.paf`. Let us suppose that a PPT file contains the following key property and value

PACK_TYPE=dip

The schematic instance may contain DIP, Dip, or dip, and the value will be still matched with the PACK_TYPE value in the PPT file.

For properties that do not have the keyword uppercasevalue (i.e. the values are to be treated case-sensitive), the comparison for the key property values between the schematic instance and the PPT can be done case-sensitively or case-insensitively. The default is case-insensitive matching. To change this to case-sensitive matching, you can select an option Perform Case Sensitive Row Match provided in the Part Table tab of Project Manager setup.

Example:

Consider PPT rows of the following type

```
:ABCD = EFGH;
 'EFGH'(1) = 'pqr1'
 'efgh'(2) = 'pqr2'
END_PART
```

If the schematic instance has the property value EFGH and you have not selected the *Perform Case Sensitive Row Match* option, Packager-XL will match the value on the schematic instance with the values in both the rows. An error message is also generated if a unique match is not found.

If you have selected the *Perform Case Sensitive Row Match* option, the match will be carried out with only the first row.

Once the row is matched, to send the PPT properties in the pst files, the algorithm is the same (the case of the values is preserved and the names are uppercased). To override these defaults, you can use the keyword uppercasevalue/preservename in the `cdsprop.paf` file.

Constructing PPTs with Case-Sensitive Values

If there are rows in a PPT that contain the key property values that differ only in case, you must specify explicit subtype names for each row.

Example:

Consider PPT rows of the following type

```
:ABCD      = EFGH   ;
 'EFGH'(1)    = 'pqr1'
 'efgh'(2)     = 'pqr2'
END_PART
```

In this example, the rows are named explicitly using 1 and 2 in the brackets following the key property values. The rows can be named also by using a unique ~<string> in each of the brackets.

Note: If an explicit naming is not done (empty brackets or ! in the brackets), the PPT will not be usable and will not loaded by any of the tools. This restriction exists because the physical part name resulting from any PPT row has to be unique after uppercasing it.

Working with Text Macros

This section describes the procedures for working with text macros in Concept HDL.

About Text Macros

You use text macros to globally replace a string of characters with another. A text macro is a text template that represents variable information that can be used in different places. When the information changes value, you need to change only the macro definition.

Text macros are used for defining global information that is needed in many places. A text macro consists of a name (identifier) and a definition.

The rules for naming a text macro are as follows:

- It can consist of letters, digits and the underscore character only.
- It can start only with a letter.
- It cannot exceed 31 characters.

A text macro definition represents a character string up to 255 characters in length.

When you run Packager-XL, it replaces occurrences of each text macro with the strings it represents. For example, the text macro CDS can represent the string Cadence Design Systems. The process of replacing the text macros with the strings of characters they represent is called text macro expansion. In the current implementation, text macros can only be used in properties on instances. Packager-XL expands the text macro placed within a property value.

How to use Text Macros

Text macros need to be identified within the property value with the ‘%’ character.

For example:

PROP1 = ‘W=%WIDTH, L=%LENGTH’

Note: The presence of the two text macros WIDTH and LENGTH in the property value is flagged with the ‘%’ character. Packager-XL only expands the identifier following the ‘%’ character. The comma marks the end of the macro identifier WIDTH and the end of string marks the end of the macro identifier LENGTH. In this example, if width was defined as 2 and length as 3, the above property would be expanded as

PROP1 = ‘W=2, L=3’

You can use a space, a comma or an end of string character to separate one macro identifier from another. If the text macro is to be immediately followed by text (that is by any character acceptable as an identifier), enclose it in quotes.

For example:

PROP1 = ‘This property value is % ’TM’ed.’

The text macro TM is identified by the quotes.

Text macros within property values can include parameters, but they cannot have embedded text macros (nested macros). If they do appear, they are ignored.

Where to Define Text Macros

There are two places to define macros

- on individual drawings using the DEFINE symbol or \parameter or \param on a hierarchical block
- in a text file

  Default Text Macro File

  No default text macro file is provided by Cadence.

  SCALD Compatibility

  The text macro file in HDL architecture is not compatible with SCALD.

Defining Text Macros on a Drawing Using the DEFINE Symbol

Text macros are defined in a drawing using DEFINE symbols. The DEFINE symbol is a part of the standard library located at <*your_inst_dir*>/ share/library/standard.

To define a text macro for a drawing, add the DEFINE symbol and use the PROPERTY command or use the *Attribute* dialog box in Concept HDL to attach the property to the

DEFINE symbol. The PROPERTY command expects a name/value pair separated by a space. The name/value pair corresponds to the identifier/definition of the macro.

For example, if you add to the DEFINE symbol CDS = ‘CADENCE DESIGN SYSTEMS’ and attach the property MY_PROP = %CDS on an instance in the schematic, Packager-XL will interpret “CDS” as the macro identifier and “CADENCE DESIGN SYSTEMS” as the macro definition and accordingly substitute CDS with CADENCE DESIGN SYSTEMS in the property value. This property will appear in the Packager-XL output file (`pstxprt.dat`) as MY_PROP = CADENCE DESIGN SYSTEMS.

There is no limit to the number of macros you can add to a DEFINE symbol or the number of DEFINE symbols you can add to a drawing. A text macro that is defined on a particular drawing using the DEFINE symbol is operative within that drawing and all other drawing (modules) within its hierarchy.

Defining Text Macros Using \PARAMETER or \PARAM

Packager-XL allows you to pass values of macros down to one level by defining macros using \parameter or \param. To define a text macro using \parameter or \param, suffix the term \parameter or \param to the property value string.

Consider the example of a hierarchical block, CNTR, which has the instances ls00 and ls04 inside it. Property LOCATIONS = U%’MY_LOC’1 is attached to ls00, and LOCATION = U%’MY_LOC’2 is attached to ls04. If the property MY_LOC = 5\parameter or MY_LOC = 5\param is attached to the hierarchical block CNTR, then running Packager-XL will cause the property LOCATION to have a values U51 and U52 for the instances ls00 and ls04, respectively.

Note: Text macro substitution takes place only when Packager-XL is run in the forward mode.

Both \param or \parameter (case insensitive) are treated as potential text macros. All properties that you define using \PARAM or \PARAMETER are written into the viewprops.prp file for the block where they are defined.

Defining a Text Macro in a File

You can define text macros that are known globally in all modules in a text macro file. When you define a global text macro in a text file, the macro cannot be overridden. A macro defined here overrides macros defined using the DEFINE symbol and using the \parameter option.

Name and Location

The reserved name for text macro file is `cdsprop.tmf`.

Note: The text macro file loaded by Packager-XL from <*your_inst_dir*>/ `cdssetup/cdsprop.tmf`. Currently, this file is empty.

The search is done in the following sequence:

- current working directory
- ~ (home)
- CDS_SITE/share
- <*your_inst_dir*>/`cdssetup`

Syntax

The text macro file contains a list of macro identifiers and associated definitions. A text macro specification is defined within one line in the file and has the following syntax:

macro_identifier = ‘macro_definition’

where macro_identifier is expressed in the VHDL name space and macro_definition is a string enclosed in quotes.

Note: The space and tab characters are always ignored outside tokens. Comments are allowed anywhere outside a token, and they begin with “#” until the end of the line.

For example, in a schematic that has two blocks within it, BLOCK1 and BLOCK2, a property MY_LOC=5\PARAMETER is attached to BLOCK1 with the ls04 instance inside it having property LOCATION=U%'MY_LOC'10. BLOCK2 has property MY_LOC=8\PARAMETER and instance ls32 inside it has property LOCATION=U%'MY_LOC'20 attached to it. A DEFINE symbol inside BLOCK2 contains the macro MY_LOC=6. Now, create the `cdsprop.tmf` file in the project directory, which has macro defined as MY_LOC=3.

When you save the schematic and package the design, global substitution of the macro takes place at all levels of the hierarchy with the macro value defined in text macro file. Thus, the property on ls04 appears as LOCATION=U310 and the property on ls32, which is inside block2, appears as LOCATION=U320.

Working with Occurrence Properties

This section describes the procedures for working with occurrence properties in Concept HDL.

About Occurrence Editing

The netlist generated for any design specified in Concept HDL has two distinct aspects:

- Connectivity
- Properties

Connectivity represents the definition of the structure of the design and is specified in Verilog using the `verilog.v` file. This file is generated by HDL-Direct every time the schematic is saved.

A property acts as a directive to the downstream tools (Packager-XL, Simulation Flow, Allegro, and so on.), and aids in controlling their functions. Properties can be attached to objects (primitive instances, pins on instances, and nets) in two modes:

- The schematic mode

  The properties attached in the schematic mode are saved in the `viewprps.prp` file.

- The occurrence mode

  The properties attached in the occurrence mode are saved in the `props.opf` file.

The schematic and occurrence modes differ in the way they treat hierarchical blocks.

Packager-XL reads both the `verilog.v` file and the `viewprps.prp` file to generate the pst* files. The properties specified on instances are stored in the `pstxprt.dat` file, and the properties specified on nets and pins are stored in the `pstxnet.dat` file.

The Schematic Mode

The schematic mode is the default mode in Concept HDL. In this mode, all properties attached to instances, nets, and pins are stored in the `viewprps.prp` file. This file is located in the schematic view (sch_1) of the cell in the design library.

The Occurrence Mode

To add properties to a specific occurrence of an object, you can use the occurrence mode of property editing in Concept HDL.

Viewing Occurrence Properties

To view occurrence edit properties:

1. Choose *Tools > Expand Design*.
2. Choose *Tools > Occurrence Edit*.

This enables occurrence editing. Occurrence properties are displayed on the schematic if they have corresponding schematic or placeholder properties. If there are no corresponding properties, you can view occurrence properties from the *Attributes* dialog box when occurrence editing is enabled.

To display occurrence property attributes with occurrence editing enabled

1. Choose *Text > Attributes*.
2. Select an object.

   The object you select is highlighted, and the *Attributes* dialog box displays the Winning values for the object.

Note: To disable occurrence editing, choose *Tools > Occurrence Edit*.

Occurrence Property File (OPF)

The occurrence property file is a database that stores properties for all objects (instances, pins, and nets). It is accessed in read-write mode from both Concept HDL and Packager- XL.

Creating an occurrence property file

The OPF is created in two ways:

- When you invoke the occurrence edit mode in Concept HDL

  Concept HDL creates an OPF, if it does not already exist (that is created by Packager-XL) and writes all the properties added in the occurrence edit mode into it.

- When you package your design

Packager-XL creates an OPF if it does not already exist (and is created by Concept HDL). Packager-XL always reads and writes properties into the OPF both in the forward and feedback mode.

If a design has been packaged at least once, the OPF is loaded by Concept HDL in the read-write mode wherein all (relevant) packaging information is accessible. You can modify by using the *Attribute* menu in the occurrence mode.

Location of the Occurrence Property File

The OPF is located in the opf view of the root cell.

Any property that is attached to an object in the occurrence mode is stored in the OPF at the root level. This is regardless of where they are located in the design hierarchy. The root cell is the design name in the `.cpm` file and can be defined through the Project Manager Setup.

Hierarchy of Occurrence Property Files

Since an OPF always resides at the root level of a design that is being edited or packaged, you can have a hierarchy of OPFs by packaging the hierarchical blocks individually.

OPFs, when arranged in a hierarchy where one OPF exists at the top level and the other OPFs exists at a lower level, are referred to as `Overlayed OPFs` and `Subdesign OPFs`, respectively. The properties in the subdesign OPFs are visible at the top level. In case of a conflict such as when a property has different values at the top and lower levels, the value at the topmost level takes precedence. This is unlike inheritance where the lowermost level always takes precedence.

Property Precedence in an OPF

There are two rules for setting the precedence of properties in an OPF.

1. If the same occurrence of an object has a schematic value and an OPF property value, the OPF property value is preferred over the schematic value.
2. If the same occurrence has multiple OPF property values stored at different levels of the hierarchy (in different OPF files), then the OPF property value at a higher level gets preference over the OPF property value at the lower level.

Inheritance of Properties in an OPF

Inheritance is used to propagate properties down in the hierarchy. An OPF handles inheritance in the same way as a schematic.

A property, whether schematic or OPF, is propagated to all lower level cells present in the hierarchy. If the same property is present in any lower level cell, the existing property at the lower level cell still gets preferred over the propagated property.

Working with Custom Text

You can attach custom text to the origin of a symbol or to objects on a schematic for displaying text that is context-specific. Custom text is different from notes and comments because the text is attached to objects and it can be context -specific. This text is made context-specific by adding custom variables to it.

Custom text is specified in the form of two strings:

- Format string

  The format string specifies the format in which the actual custom text is to be displayed. It may contain custom variables and environment variables.

- Display string

  The display string contains the substituted values of the custom and environment variables. This is the string that actually appears on the schematic.

Given below is an example of using custom variables:

Using two variables of Concept HDL <CON_PAGE_NUM> and <CON_TOTAL_PAGES>, specify the format string as:

```
This is page <CON_PAGE_NUM> of <CON_TOTAL_PAGES>
```

After substituting the values of the custom variables, the display string is:

```
This is page 1 of 10
```

Types of Custom Variables

There are two types of custom variables:

- Defined within Concept HDL
- User-Defined

The values for custom variables that are defined within Concept HDL are supplied by Concept HDL itself. The case of the values is the same as used by Concept HDL. For example, in Concept HDL , the design names and library names are in lowercase. So, the value of an

inbuilt custom variable, say CON_DESIGN_LIB, would also be in lowercase. You cannot change the values of these variables.

For variables that you define, you need to provide the values.

Custom variables make the plots of cross-referenced schematics more illustrative and easy to use. Concept HDL provides some pre-defined variables whose values are substituted by cross-referencer. These are available in cross-referenced schematics. Custom text is not visible in HPF plots.

Inbuilt Concept HDL Variables

Drawing-Specific Variables

The values of these variables are different across different drawings.

- CON_DESIGN_LIB

  Name of the design library

- CON_DESIGN_NAME

  Name of the design

- CON_DESIGN_VIEW

  Currently open view of the design. It is one of the schematic views

- CON_PAGE_NUM

  Current page of the drawing in a cell. It is different for each page of the drawing.

  This variable does not take into account the pages in all the cells used in a hierarchical design. For more information, see The CON_PAGE_NUM and CON_TOTAL_PAGES Custom Text Variables on page 359.

- CON_LAST_MODIFIED

  The date and time when the current drawing was last modified.

- CON_TOTAL_PAGES

  Total number of pages in a cell. The value of `CON_TOTAL_PAGES` will be the value of the highest page number you have assigned for a page in the cell. For example, if you have four pages in a cell with numbers 1, 2, 3, and 6, the value of `CON_TOTAL_PAGES` is 6.

This variable does not take into account the pages in all the cells used in a hierarchical design. For more information, see The CON_PAGE_NUM and CON_TOTAL_PAGES Custom Text Variables on page 359.

- CURRENT_DESIGN_SHEET

  Current page of the drawing in a hierarchical design. For more information, see The CURRENT_DESIGN_SHEET and TOTAL_DESIGN_SHEETS Custom Text Variables on page 360.

- TOTAL_DESIGN_SHEETS

  Total number of pages in a hierarchical design. For more information, see The CURRENT_DESIGN_SHEET and TOTAL_DESIGN_SHEETS Custom Text Variables on page 360.

Global Variables

The value of these variables is constant across all pages of the schematic.

- CON_ROOT_LIB

  Library to which the root design belongs

- CON_ROOT_NAME

  Name of the root design

- CON_ROOT_VIEW

  View of the root design

- CON_PROJ_NAME

  Name of the current project file

Parent Variables

The value of these variables is set when you descend into the hierarchy or edit using the canonical name of the design. These variables are not set for root designs.

- CON_PARENT_NAME

  Name of the parent design

- CON_PARENT_CNAME

Canonical name of the parent design. The value is substituted in accordance with the Concept HDL option of displaying lib, cell, view.

- CON_PARENT_LIB

  Name of the library to which the parent of the current design belongs

- CON_PARENT_VIEW

  View of the parent design

CRef Variables

The value of these variables is set in the `schcref_1` view when the design is cross-referenced with the option *Create Flattened Schematic*.

- CREF_TO_LIST

  Defines where the pages for the blocks are located in the cross-referenced flattened design.

- CREF_FROM_LIST

  Defines where the pages are coming from in a flattened design

- CREF_ORIG_DESIGN_NAME

  Defines the original design name

- CREF_ORIG_PAGE

  Defines the original page number

- CREF_ORIG_VIEW

  Defines the view of the original design

User-Defined Variables

You can define new custom variables for storing and displaying information such as COMPANY NAME and AUTHOR. Concept HDL writes the user-defined custom variables for a project in the project file (`.cpm`). For more information on defining new custom variables, see Defining New Custom Variables on page 243.

Using Custom Text

It is recommended that you add custom text to the symbol of a page border. To add custom text to symbols, attach it to the origin. The symbol just shows the format string. When the page border is instantiated, the values of custom variables are substituted. For example, if you have the following custom text on the page border symbol:

```
Page <CON_PAGE_NUM>
```

When the page border gets instantiated, the custom variable `CON_PAGE_NUM` would take its actual value on each page. For example, Page 1 or Page 2.

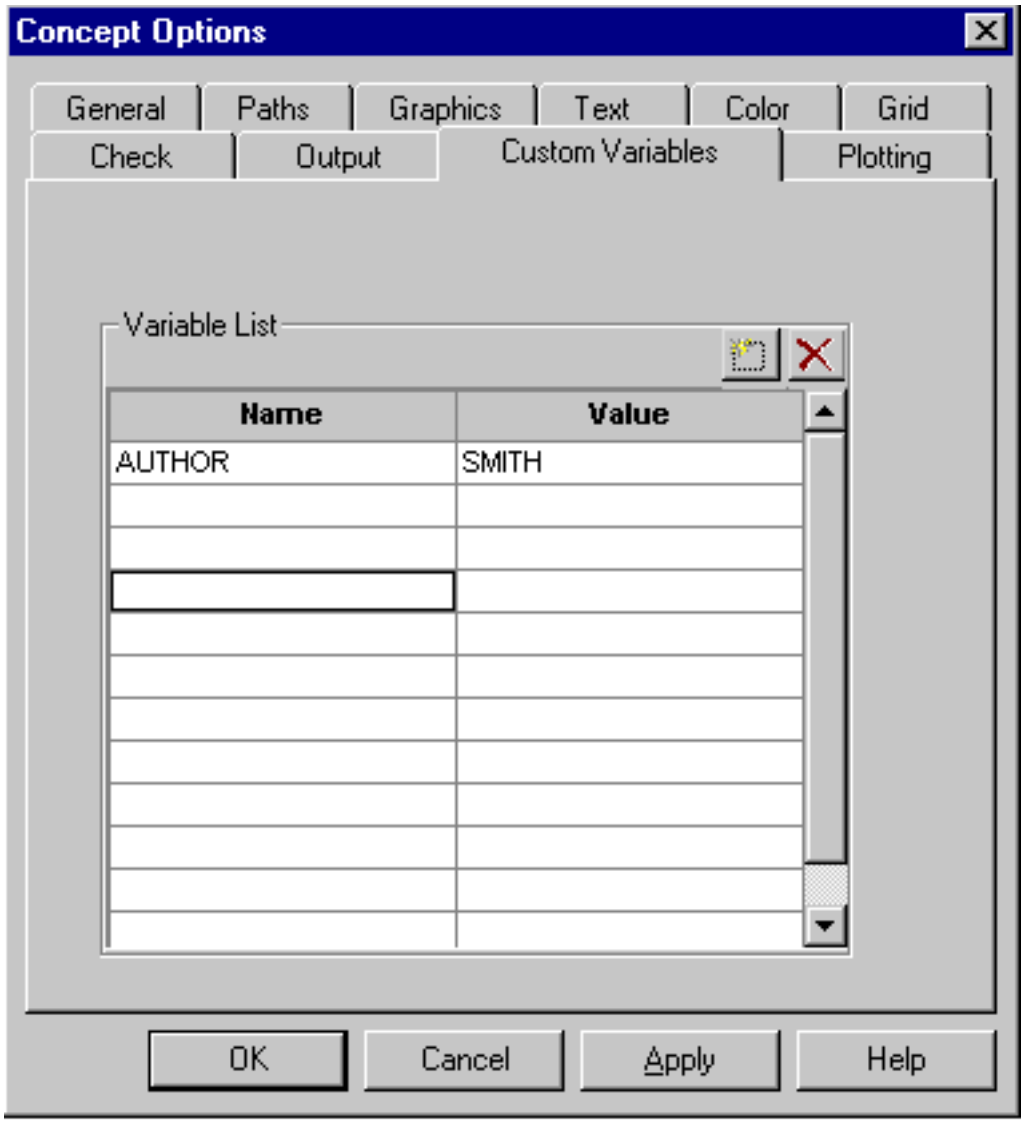
Defining New Custom Variables

You can define your own variables to use in a design project. These variables can be defined using the *Concept Options* dialog box.

To define new variables:

1. Choose *Tools* > *Options*.

The *Concept Options* dialog box appears.



2. Click on the *Custom Variables* tab.
3. Enter the name of the variable.
4. Enter the value of the variable.

   The value is constant for all pages of the design.

   For example, you can enter AUTHOR as the name of the variable and SMITH as the value.

   You can define more variables by clicking on the button.

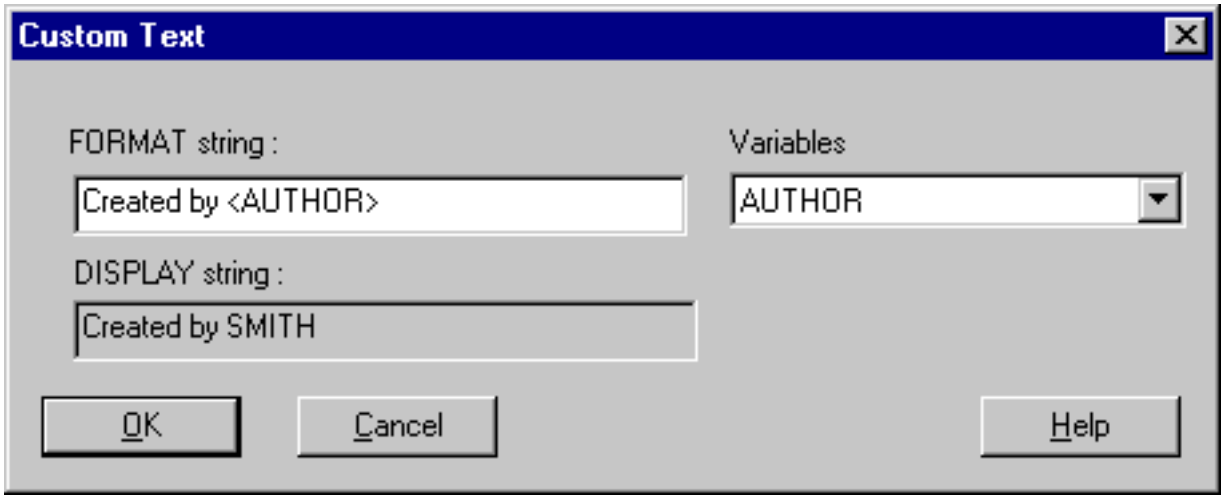
5. Click *Apply*.
6. Click *OK*.

   Concept HDL adds the custom variable to your design project.

Adding Custom Text

1. Choose *Text > Custom Text*.

   The Custom Text dialog box appears.



2. Enter the format string for the custom text.
3. Select the variable from the Variables drop-down list or type your own custom variable. You can also add an environment variable.

   The variable is added to the format string. The *Display* string field displays this line with the current value of the variable.

   If you want to add an environment variable to the format string, precede it with a $ sign. The current value of the environment variable is displayed in the Display string field.
4. Click *OK*.

   The custom text gets attached to the cursor.
5. Click on an object to attach custom text to it.

   Note: If you are adding a custom text on a symbol, click on the origin of the symbol to attach the custom text.
6. Click again to place the custom text at the location where you want it to be displayed.

7. Right click and select the *Done* option.

Note: You can add multiple custom text to the same object.

Case-Sensitivity in Custom Text

If you add a variable in the Unix environment, the case of the variable is very important. You can set the option of storing input in uppercase in Concept HDL using *Concept Options > Text* and enabling the check box *Upper-case Input*. While setting a variable, if this check box is checked, the variable name is stored in uppercase. So, while adding the variable to custom text, you would have to enter the name of the environment variable in uppercase.

Modifying Custom Text

1. Choose *Text > Change*.
2. Click on the custom text you want to modify.

   The *Custom Text* dialog box appears with the format string highlighted.
3. Edit the text in the *FORMAT* string field.
4. Click *OK*.

   The custom text is modified.

Note: The operations like delete, move, copy, rotate and spin that can be performed on properties can be done on custom text also.

11

Working with Electrical Constraints

A constraint is a user-defined requirement applied to a net or pin-pair in a design. Electrical constraints (ECs) govern the electrical behavior of a net or pin-pair in a design. For example, you can capture a constraint to define the maximum voltage overshoot tolerated by a net and capture the minimum first switch delay for a driver-receiver pin-pair in your design.

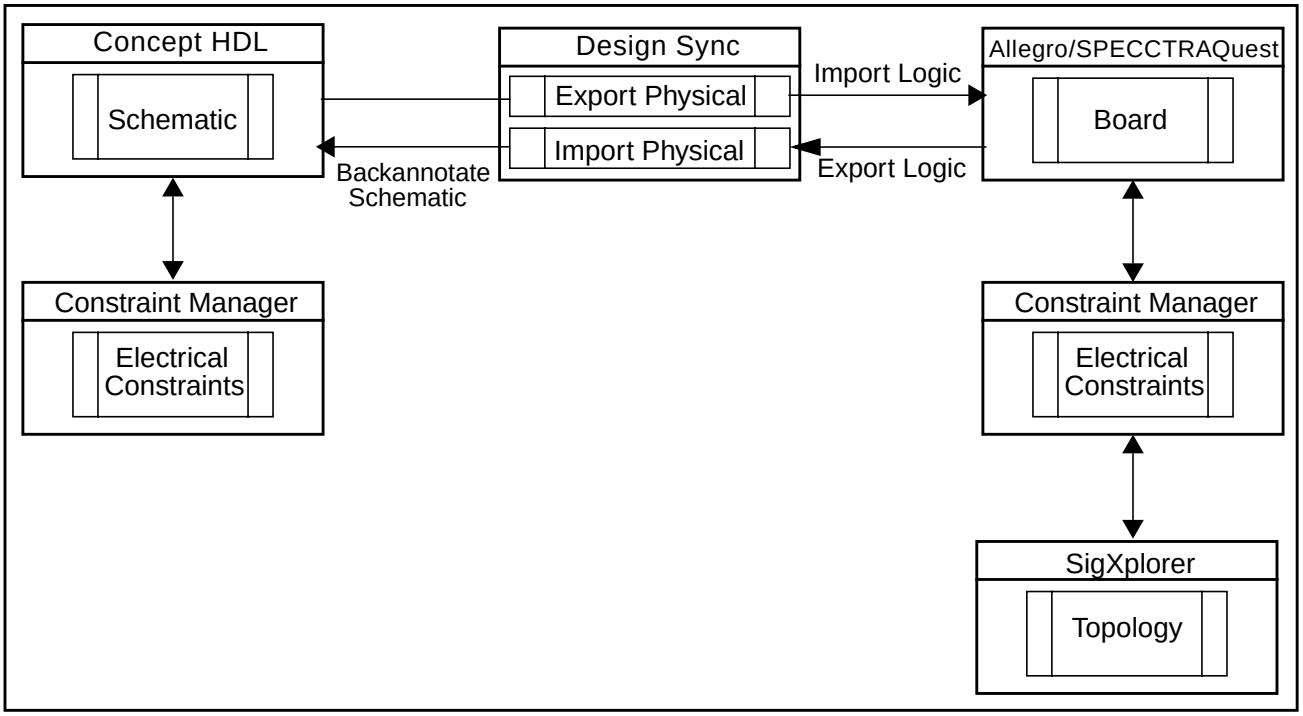
You can use Constraint Manager with Concept HDL to capture and manage electrical constraints as you implement logic. The changes that you make to constraint information in Constraint Manager are displayed in Concept HDL. Similarly, the changes that you make to constraint information in Concept HDL are displayed in Constraint Manager.

Cadence recommends that you capture constraints in Constraint Manager. This is because Constraint Manager has the following features:

- It provides a spread-sheet based user interface that allows you to quickly capture, modify, and delete constraints.
- It supports syntax checking for all constraints.
- It supports constraint inheritance. The constraints captured on a higher level object in a design is inherited by the lower level objects in the design. You can capture a constraint on a lower level design object to override the same constraint that is inherited from a higher level design object.
- It lets you create Electrical Constraint Sets (ECSets)—a collection of electrical constraints that define a particular design requirement— and assign them to objects on which you want to capture the same set of constraints. For example, you can create an ECSet to define the default timing and noise tolerance for a net. An Electrical Constraint Set applies to an individual net although other nets may contribute to the measure of the constraint (for example, to crosstalk).

  Modifying an ECSet automatically applies to all objects on which the ECSet is assigned.
- It lets you generate reports on constraints captured in the schematic and the board.

Concurrent Capture of Constraints in Schematic and Board



Your workflow can also support the concurrent capture of constraints in both the schematic and the board. Design Synchronization processes resolve constraint modifications in both directions—schematic to board and board to schematic. For more information, see Synchronizing Electrical Constraints between Schematic and Board on page 302. For more information on capturing constraints on the board, see the documentation for *Allegro/APD*, *SPECCTRAQuest*, and *Constraint Manager*.

What's New in Constraint Management in Concept HDL 14.2

In this release, the integration between Concept HDL 14.2 and Constraint Manager has been improved to provide the following features:

- Support for synchronization of electrical constraints in Constraint Manager and Concept HDL

  In this release, the electrical constraints that you capture in Constraint Manager are synchronized in Concept HDL and vice versa.

The constraints that you capture in Constraint Manager are updated in Concept HDL when you choose *Tools > Constraints > Update Schematic* in Concept HDL. You can view the constraints using the *Attributes* dialog box in Concept HDL. The constraints that you capture in Constraint Manager will not be automatically visible on the schematic in Concept HDL. To make a constraint visible in Concept HDL, add a placeholder for the constraint on the schematic. For more information, see Displaying Electrical Constraints Captured in Constraint Manager on the Schematic on page 291.

The constraints that you capture in Concept HDL are displayed in Constraint Manager when you start Constraint Manager from Concept HDL.

In Concept HDL 14.0 and 14.1, if you capture a constraint in Concept HDL, and start Constraint Manager from Concept HDL, the constraint is not displayed in Constraint Manager. Similarly, a constraint captured in Constraint Manager is not displayed in Concept HDL.

- Improved support in the Design Synchronization flow for importing and exporting constraints if you are using Constraint Manager with Concept HDL

  - In this release, when you run Import Physical and run backannotation, the constraints existing in both Constraint Manager and Concept HDL are updated and the constraint information in both Constraint Manager and Concept HDL are automatically synchronized.

    In Concept HDL 14.0 and 14.1, if you run Import Physical and perform backannotation, the constraints existing in Concept HDL and Constraint Manager are updated separately. The changes to constraints in Constraint Manager after backannotation were not reflected in Concept HDL. Similarly, the changes to constraints in Concept HDL after backannotation were not reflected in Constraint Manager.

  - Support for backannotation from *Import Physical* and *Export Physical* dialog boxes

    You can select the *Backannotate Schematic* check box in the *Import Physical* and *Export Physical* dialog boxes to backannotate changes in electrical constraint properties as well as other properties that you have made in the board, to the schematic. This saves you the need to run *Tools > Back Annotate* in Concept HDL to backannotate the changes made in the board into the schematic. You can even backannotate the changes in the Occurrence Edit mode.

  - You no longer have to use the *Property Flow Setup* dialog box to specify that a particular electrical constraint property has to be transferred from the board to the schematic. All electrical constraint properties are automatically transferred from the board to the schematic when you run Import Physical in Concept HDL 14.2.

In Concept HDL 14.0 and 14.1, you had to select the *Transfer* check box next to each electrical constraint property listed in the *Property Flow Setup* dialog box to indicate that the electrical constraint properties have to be transferred from board to schematic when you run Import Physical.

- In this release, the Design Differences tool supports display of a report of differences in electrical constraints information between schematic and board. The differences are displayed in the *Constraints Differences-Logical* and *Constraints Differences-Physical* windows. For more information, see Viewing Constraint Differences between the Schematic and the Board on page 297.

- Usability enhancements in using Constraint Manager with Concept HDL

  - Support for adding placeholders for electrical constraints in the schematic

    The constraints that you capture in Constraint Manager are updated in Concept HDL when you choose *Tools* > *Constraints* > *Update Schematic* in Concept HDL. You can view the constraints using the *Attributes* dialog box in Concept HDL. The constraints that you capture in Constraint Manager will not be automatically visible on the schematic in Concept HDL. You can make a constraint visible in Concept HDL by adding a placeholder for the constraint on the schematic. Concept HDL allows you to quickly add placeholders for constraints in the schematic using the *Attributes* dialog box. For more information, see Displaying Electrical Constraints Captured in Constraint Manager on the Schematic on page 291.

  - Syntax checking of electrical constraint properties in the schematic when you start Constraint Manager from Concept HDL or run Export Physical

    Errors in the syntax of the value for electrical constraint properties on the schematic are displayed in the `concept2cm.log` file when you start Constraint Manager from Concept HDL or run Export Physical. For more information on the `concept2cm.log` file, see concept2cm.log on page 315.

  - *Tools* > *Back Annotate* in Concept HDL now supports backannotation of electrical constraints.

  - Before you package your design in Concept HDL, logical net names are displayed in Constraint Manager. After you package your design, the Constraint Manager UI is automatically refreshed to display the physical net names. For more information, see How Nets Are Displayed in Constraint Manager on page 314.

  - If you make any connectivity changes in Concept HDL when Constraint Manager is running, they become visible in Constraint Manager when you save the design in Concept HDL. When you save any changes in Constraint Manager, you can view the changes using the *Attributes* dialog box in the Expanded mode in Concept HDL.

In Concept HDL 14.0 and 14.1, after you save connectivity changes in Concept HDL you must re-expand (choose *Tools* > *Expand Design*) the design in Concept HDL to view the changes in Constraint Manager. When you save any changes in Constraint Manager, you can view the changes using the *Attributes* dialog box in the Expanded mode in Concept HDL only if you re-expand the design.

Using Designs from Previous Versions of Concept HDL

The following sections describe the tasks you have to perform if you want to use a design that you have created in Concept HDL 14.1 or a previous version in Concept HDL 14.2.

- Using Concept HDL 14.0 or 14.1 Designs in Concept HDL 14.2 on page 251
- Using Concept HDL 13.6 designs in Concept HDL 14.2 on page 253

Using Concept HDL 14.0 or 14.1 Designs in Concept HDL 14.2

A design that you have created in Concept HDL 14.0 or 14.1 might have electrical constraint properties. The following sections describe the tasks you have to perform if you want to use the design in Concept HDL 14.2.

- If you have used Constraint Manager with Concept HDL 14.0 or 14.1 on page 251
- If you did not use Constraint Manager with Concept HDL 14.0 or 14.1 but want to use Constraint Manager with Concept HDL 14.2 on page 252
- If you did not use Constraint Manager with Concept HDL 14.0 or 14.1 and do not want to use Constraint Manager with Concept HDL 14.2 on page 253

If you have used Constraint Manager with Concept HDL 14.0 or 14.1

If your Concept HDL 14.0 or 14.1 design has electrical constraint properties and you were using Constraint Manager with Concept HDL 14.0 or 14.1, do the following before using the design in Concept HDL 14.2:

1. Ensure that you are using Allegro and SPECCTRAQuest 14.2.
2. Open the design in Concept HDL 14.2.
3. Choose *Tools* > *Constraints* > *Update Schematic*.

   Running *Update Schematic* does the following:

- Resolves differences in the value of constraints stored in Constraint Manager 14.0 or 14.1 and the corresponding electrical constraint properties in the Concept HDL 14.0 or 14.1 schematic. The values that exist in Constraint Manager 14.0 or 14.1 override the value of electrical constraint properties in the schematic.

  For example, if the value for maximum crosstalk in the *Estimated XTalk* worksheet on a net `INT` in Constraint Manager 14.0 or 14.1 is `0.4000:0.4000` and the `MAX_XTALK` constraint property on the net `INT` in the Concept HDL 14.0 or 14.1 schematic has the value `0.5000:0.5000`, the value of the `MAX_XTALK` property on the net `INT` in the schematic is updated to `0.4000:0.4000` in Concept HDL 14.2.

- If a placeholder existed on the Concept HDL 14.0 or 14.1 schematic for an electrical constraint captured in Constraint Manager 14.0 or 14.1, the electrical constraint gets backannotated to the schematic. You can now view the latest constraints information on the schematic in Concept HDL 14.2.

- If a placeholder did not exist on the Concept HDL 14.0 or 14.1 schematic for an electrical constraint captured in Constraint Manager 14.0 or 14.1, the electrical constraints captured in Constraint Manager 14.0 or 14.1 become visible in the *Attributes* dialog box with *Visibility* set to *None*, if you are in the Expanded or Occurrence Edit mode in Concept HDL 14.2.

If you have used Constraint Manager with Concept HDL 14.0 or 14.1, in Concept HDL 14.2, you are in the Constraint Manager enabled flow. For more information, see Constraint Manager Enabled Flow on page 263.

Important

If you are in the Constraint Manager enabled flow:

- You must not use Concept HDL 14.2 with Allegro or SPECCTRAQuest 14.1 or earlier versions

- You must not use Concept HDL 14.1 or a previous version with Allegro or SPECCTRAQuest 14.2

If you did not use Constraint Manager with Concept HDL 14.0 or 14.1 but want to use Constraint Manager with Concept HDL 14.2

No changes are required if you did not use Constraint Manager with Concept HDL 14.0 or 14.1 but want to use Constraint Manager with Concept HDL 14.2.

You can start Constraint Manager from Concept HDL 14.2 to capture and manage electrical constraints. All the electrical constraints present in the schematic are pushed to Constraint Manager and you can view the constraints in Constraint Manager.

Once you start using Constraint Manager with Concept HDL 14.2, you are in the Constraint Manager enabled flow. For more information, see Constraint Manager Enabled Flow on page 263.

Important

If you are in the Constraint Manager enabled flow:

- You must not use Concept HDL 14.2 with Allegro or SPECCTRAQuest 14.1 or earlier versions
- You must not use Concept HDL 14.1 or a previous version with Allegro or SPECCTRAQuest 14.2

For more information, see Why Constraint Manager with Concept HDL 14.2 is not compatible with Allegro or SPECCTRAQuest 14.1 or previous versions on page 272.

If you did not use Constraint Manager with Concept HDL 14.0 or 14.1 and do not want to use Constraint Manager with Concept HDL 14.2

No changes are required if you did not use Constraint Manager with Concept HDL 14.0 or 14.1 and do not want to use Constraint Manager with Concept HDL 14.2.

Note: If you are not using Constraint Manager with Concept HDL 14.2, you are using the traditional flow. For more information on the traditional flow, see Traditional Flow on page 260. If you later on want to use Constraint Manager with Concept HDL 14.2, perform the steps described in Moving from Traditional Flow to Constraint Manager Enabled Flow on page 267.

Using Concept HDL 13.6 designs in Concept HDL 14.2

A design that you have created in Concept HDL 13.6 might have electrical constraint properties. The following sections describe the tasks you have to perform if you want to use the design in Concept HDL 14.2.

- If you open a Concept HDL 13.6 design in Concept HDL 14.2 and want to use Constraint Manager with Concept HDL 14.2 on page 254
- If you open a Concept HDL 13.6 design in Concept HDL 14.2 and do not want to use Constraint Manager with Concept HDL 14.2 on page 258

If you open a Concept HDL 13.6 design in Concept HDL 14.2 and want to use Constraint Manager with Concept HDL 14.2

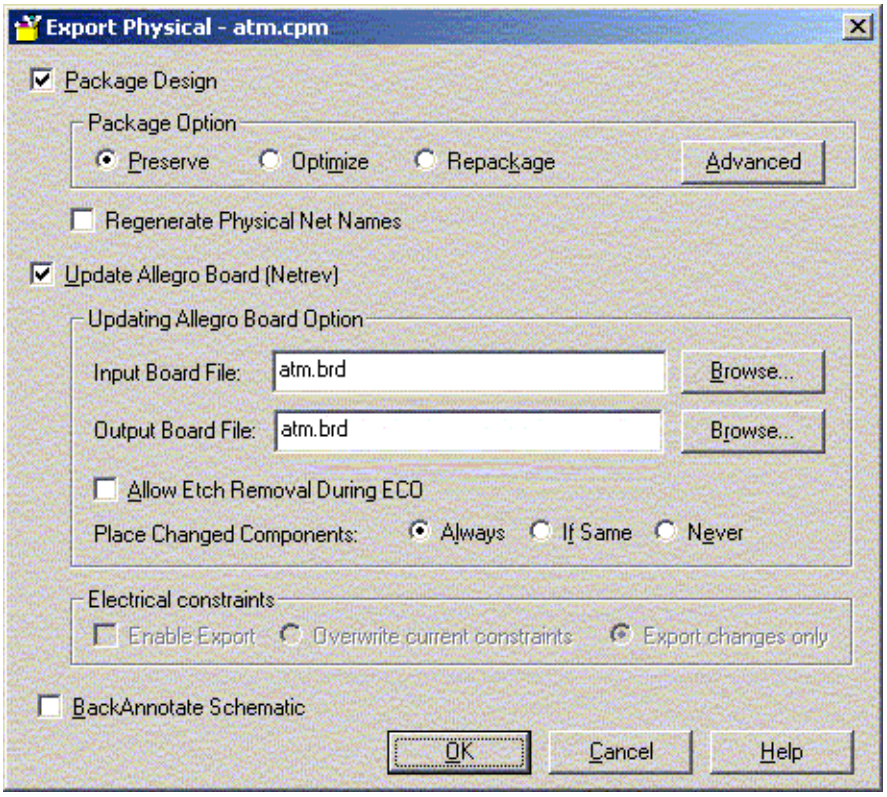
The syntax of the values for some electrical constraint properties has changed from the 14.0 release. Also, the `DELAY_RULE` and `MATCHED_DELAY` electrical constraint properties used in Concept HDL 13.6 are named as `PROPAGATION_DELAY` and `RELATIVE_PROPAGATION_DELAY` respectively from the 14.0 release.

If you open a Concept HDL 13.6 design that has electrical constraint properties in Concept HDL 14.2, you must do the following before starting Constraint Manager from Concept HDL 14.2 to ensure that the name and syntax of the value for electrical constraint properties are updated to the new format. If you do not do this, you cannot view the constraints in Constraint Manager because the constraints in the schematic are in the old format.

Note: Cadence recommends that you synchronize your schematic and board in Concept HDL 13.6 before you use your design in Concept HDL 14.2.

1. Ensure that you are using Allegro and SPECCTRAQuest 14.2.
2. Open the design in Concept HDL 14.2.
3. Choose *File > Save Hierarchy* or run the `hier_write` console window command in Concept HDL.
4. Choose *File > Export Physical* in Concept HDL.

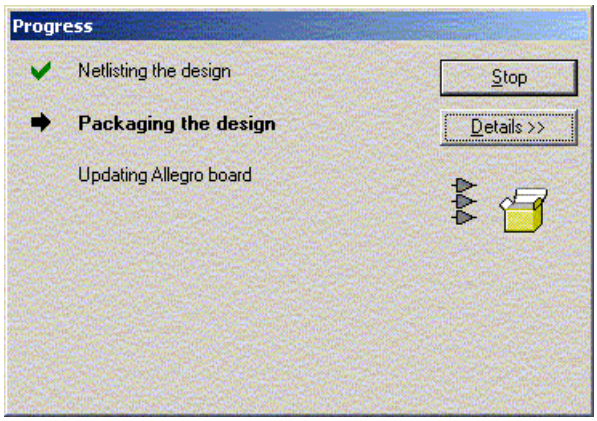
The *Export Physical* dialog box appears.



5. Select the *Package Design* check box.
6. Select the *Update Allegro Board (Netrev)* check box.

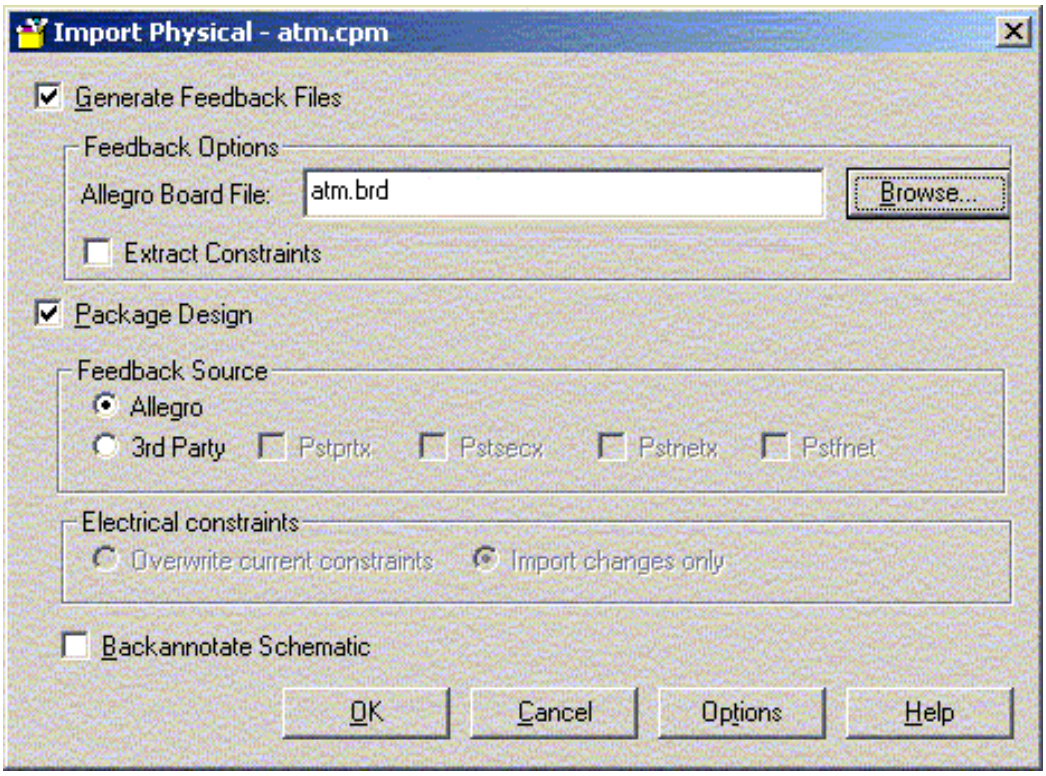
   When you run Export Physical, `Netrev` converts the syntax of the value for electrical constraint properties that are in the old format to the new format.
7. Click *OK*.

The *Progress* dialog box appears, displaying the progress of the Export Physical process.



8. Choose *File > Import Physical* in Concept HDL.

The *Import Physical* dialog box appears.



9. Select the *Generate Feedback Files* check box.

10. Select the *Extract Constraints* check box.

The following message box appears:

Design Sync
Once the Constraint Manager Flow is enabled, for this design, you cannot go back to the traditional flow. Continue extracting constraints ?
Yes
No

For more information on the traditional flow and the Constraint Manager enabled flow, see Traditional Flow on page 260 and Constraint Manager Enabled Flow on page 263.

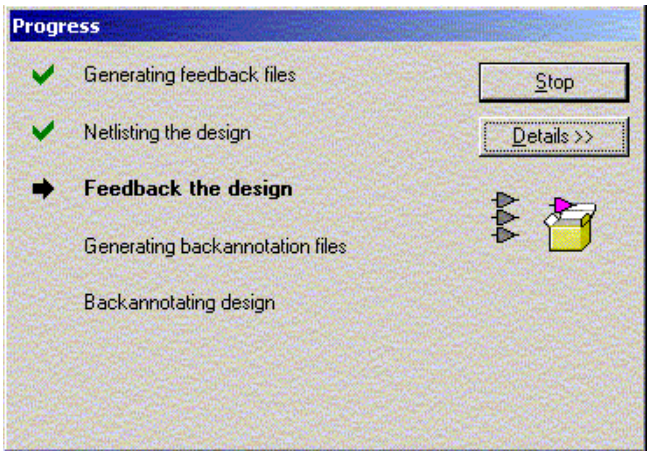
11. Click *Yes*.

12. Select the *Package Design* check box and the *Allegro* option.

13. Select the *Backannotate Schematic* check box.

14. Click *OK*.

The *Progress* dialog box appears, displaying the progress of the Import Physical process.



The syntax of the value for electrical constraint properties on the schematic are updated to the new format.

Note: The syntax of the value for electrical constraint properties on the schematic for reuse or replicated blocks in the design are not updated to the new format in Concept HDL 14.2. This will not affect the design flow between Concept HDL and Allegro because the properties with the new syntax are stored as occurrence properties and can be viewed using the *Attributes* dialog box in the Expanded or Occurrence Edit mode in Concept HDL 14.2.

The DELAY_RULE and MATCHED_DELAY electrical constraint properties, if any, on the schematic are renamed as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively on the schematic.

Note: The DELAY_RULE and MATCHED_DELAY electrical constraint properties, if any, on the schematic for reuse or replicated blocks in the design are not renamed as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively. This will not affect the design flow between Concept HDL and Allegro because the new property names and their values are stored as occurrence properties and can be viewed using the *Attributes* dialog box in the Expanded or Occurrence Edit mode in Concept HDL 14.2.

In Constraint Manager, the PROPAGATION_DELAY constraint is stored in the *Min/Max Propagation Delays* worksheet and the RELATIVE_PROPAGATION_DELAY constraint is stored in the *Relative Propagation Delay* worksheet. For information on the cell in which a constraint is displayed, see Appendix B, “Property Mapping” of the *Constraint Manager Design Guide*.

15. Choose *Tools* > *Constraints* > *Edit* to start Constraint Manager.

All the electrical constraints present in the schematic are pushed to Constraint Manager and you can view the constraints in Constraint Manager.

Once you start using Constraint Manager with Concept HDL 14.2, you are in the Constraint Manager enabled flow. For more information, see Constraint Manager Enabled Flow on page 263.

Important

If you are in the Constraint Manager enabled flow:

- You must not use Concept HDL 14.2 with Allegro or SPECCTRAQuest 14.1 or earlier versions
- You must not use Concept HDL 14.1 or a previous version with Allegro or SPECCTRAQuest 14.2

For more information, see Why Constraint Manager with Concept HDL 14.2 is not compatible with Allegro or SPECCTRAQuest 14.1 or previous versions on page 272.

If you open a Concept HDL 13.6 design in Concept HDL 14.2 and do not want to use Constraint Manager with Concept HDL 14.2

The syntax of the values for some electrical constraint properties has changed from the 14.0 release. Also, the DELAY_RULE and MATCHED_DELAY electrical constraint properties used

in Concept HDL 13.6 are named as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively from the 14.0 release.

Do the following to use a Concept HDL 13.6 design that has electrical constraint properties in Concept HDL 14.2:

Note: Cadence recommends that you synchronize your schematic and board in Concept HDL 13.6 before you use your design in Concept HDL 14.2.

1. Ensure that you are using Allegro and SPECCTRAQuest 14.2.
2. Open the design in Concept HDL 14.2.
3. Choose *File > Save Hierarchy* or run the `hier_write` console window command in Concept HDL.
4. Add the following properties in the `pxlBA.txt` file located in the `physical` view of the root design:
   - DELAY_RULE
   - MATCHED_DELAY
   - PROPAGATION_DELAY
   - RELATIVE_PROPAGATION_DELAY

   This is required for the DELAY_RULE and MATCHED_DELAY electrical constraint properties on the schematic to be renamed as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively when you run Import Physical in Concept HDL 14.2.
5. Choose *Tools > Setup* in Project Manager.

   The *Project Setup* dialog box appears.
6. Select the *Tools* tab.
7. Click *Packager-XL Setup*.

   The Project Setup dialog box appears.
8. Select the *From Layout* tab.
9. Select the *Net* check box in the *Annotate* group box and click *OK*.

   This is required for the DELAY_RULE and MATCHED_DELAY electrical constraint properties on the schematic to be renamed as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively when you backannotate the schematic in Concept HDL 14.2.

When you run Export Physical in Concept HDL 14.2, the syntax of the values for all electrical constraints on the board are updated to the new format.

When you run Import Physical in Concept HDL 14.2 and run backannotation:

- The DELAY_RULE and MATCHED_DELAY electrical constraint properties on the schematic are renamed as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively.

  Note: The DELAY_RULE and MATCHED_DELAY electrical constraint properties, if any, on the schematic for reuse or replicated blocks in the design are not renamed as PROPAGATION_DELAY and RELATIVE_PROPAGATION_DELAY respectively. This will not affect the design flow between Concept HDL and Allegro because the new property names and their values are stored as occurrence properties and can be viewed using the *Attributes* dialog box in the Expanded or Occurrence Edit mode in Concept HDL 14.2.

- The syntax of the value for electrical constraint properties on the schematic are updated to the new format.

  Note: The syntax of the value for electrical constraint properties on the schematic for reuse or replicated blocks in the design are not updated to the new format in Concept HDL 14.2. This will not affect the design flow between Concept HDL and Allegro because the properties with the new syntax are stored as occurrence properties and can be viewed using the *Attributes* dialog box in the Expanded or Occurrence Edit mode in Concept HDL 14.2.

Note: If you are not using Constraint Manager with Concept HDL 14.2, you are using the traditional flow. For more information on the traditional flow, see Traditional Flow on page 260. If you later on want to use Constraint Manager with Concept HDL 14.2, perform the steps described in Moving from Traditional Flow to Constraint Manager Enabled Flow on page 267.

Traditional Flow

If you are not using Constraint Manager with Concept HDL 14.2, you are using the traditional flow.

Note: If you later on want to use Constraint Manager with Concept HDL 14.2, perform the steps described in Moving from Traditional Flow to Constraint Manager Enabled Flow on page 267.

The Traditional Flow figure on page 262 displays the traditional flow. In this flow the following three files are created by Packager-XL when you run Export Physical (with *Package Design* check box selected):

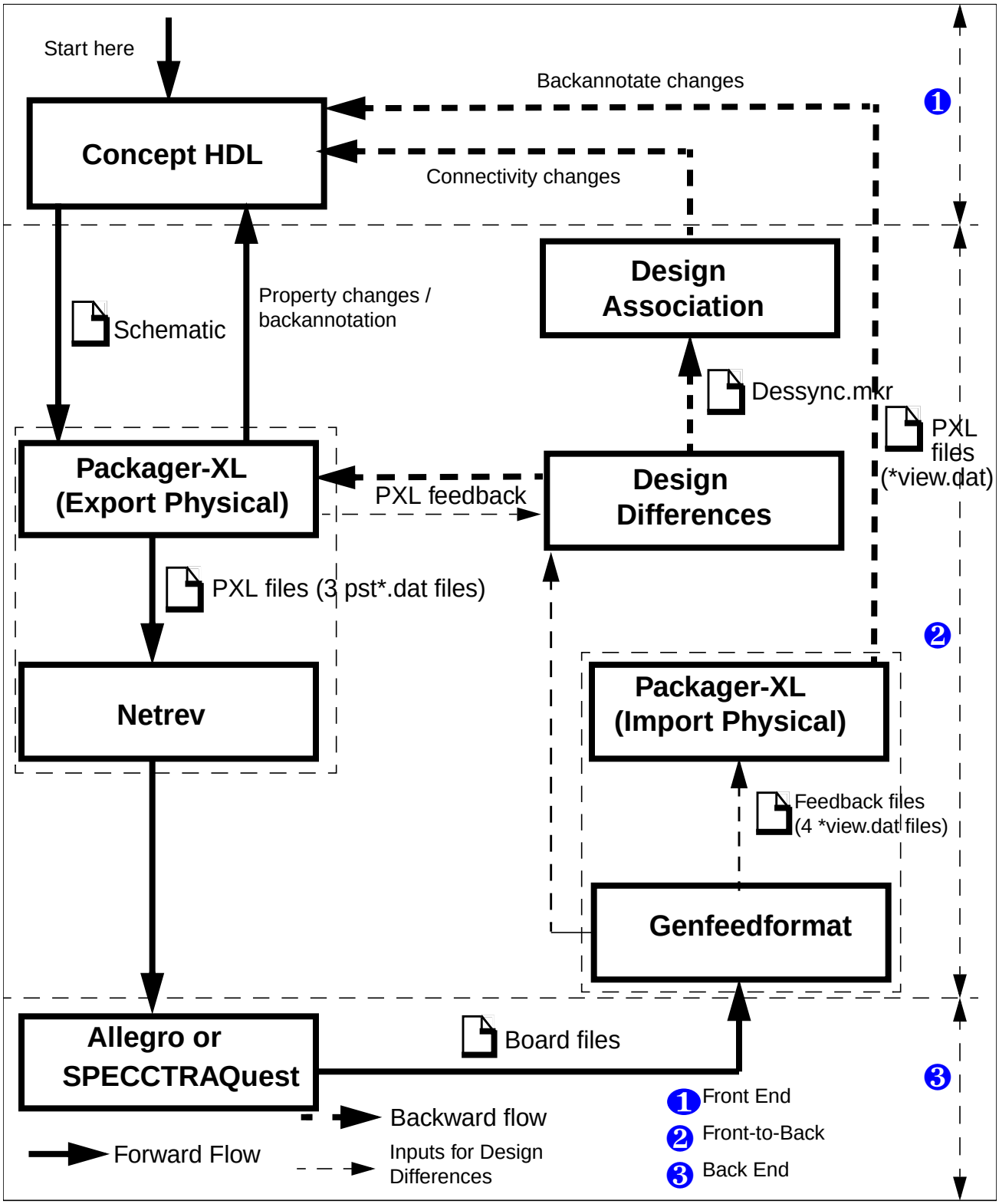
- **pstchip.dat—**Contains a physical description for each physical part used in your design. Packager-XL extracts this physical description from chips files, ptf files, and properties on schematic instances. This file contains a description of only the physical parts used in the design.
- **pstxprt.dat**—Lists each reference designator and the sections assigned to it.
- **pstxnet.dat**—The `pstxnet.dat` file is the connectivity file. This file lists each net, its properties (including electrical constraint properties), its attached nodes, and node properties. The list is ordered by physical net name and contains all net properties and the logic-to-physical binding of nets and nodes.

These files are used by `Netrev` to create or update the board. You can run Netrev from Export Physical or Allegro (*File > Import > Logic*).

You can make changes in Allegro and then feed back the changes in the board to the schematic by generating the feedback files using genfeedformat. You can run genfeedformat from Import Physical or Allegro. The following files are generated by genfeedformat:

- **pinview.dat**—Contains connectivity and pin instance properties info generated by Allegro
- **netview.dat**—Contains property information for the nets generated by Allegro
- **funcview.dat**—Contains property information for the schematic instances generated by Allegro
- **compview.dat**—Contains property information for the component instances generated by Allegro

Figure 11-1 Traditional Flow



You can now use the feedback files to synchronize the schematic and the board by doing one of the following:

- Choose *Tools > Back Annotate* in Concept HDL to backannotate all the changes in the board to the schematic.
- Use the Design Differences (VDD) and Design Association (DA) tools to resolve individual connectivity and property differences between the schematic and the board. Use VDD to update the property differences either to the board or to the schematic. When you run VDD, it displays differences in properties between the schematic and the board in multiple windows. Use DA to update the connectivity changes made in the board to the schematic. DA uses a file generated by VDD named `dessync.mkr` (which captures connectivity information) to guide you in updating the schematic.

For more information on the traditional flow and the tools used in the flow, see the *Design Synchronization and Packaging User Guide* and the *Packager-XL Reference*.

Files Needed for Board Layout in Traditional Flow

You may be creating the schematic yourself and getting the board layout done by a board designer. In such cases, after you package your design, you may be sending only the `pst*.dat` files to the board designer. If you are using the traditional flow you must send the following three files to the board designer:

- pstchip.dat
- pstrxprt.dat
- pstxnet.dat

Constraint Manager Enabled Flow

If you are using Constraint Manager with Concept HDL 14.2, you are using the Constraint Manager enabled flow.

Important

If you are using the Constraint Manager enabled flow:

- You must not use Concept HDL 14.2 with Allegro or SPECCTRAQuest 14.1 or earlier versions.
- You must not use Concept HDL 14.1 or a previous version with Allegro or SPECCTRAQuest 14.2

The Constraint Manager Enabled Flow figure on page 265 displays the Constraint Manager enabled flow. In this flow, Constraint Manager stores information about electrical constraints in a new view named `constraints` under the root design. The `constraints` view contains a file named <*root_design_name*>.dcf, which contains a snapshot of electrical constraint information in the design.

In the Constraint Manager enabled flow, Packager-XL creates five `pst*.dat` files when you run Export Physical (with *Package Design* and *Update Allegro Board (Netrev)* check boxes selected). These include the three files generated in the traditional flow (`pstchip.dat, pstxprt.dat`, and `pstxnet.dat`) and the following two files:

- **pstcmdb.dat**—Contains information about the electrical constraints currently present in the design. This file is a copy of the <*root_design_name*>.dcf file in the `constraints` view.

  Note: In the traditional flow, information about electrical constraints on the schematic are stored in the `pstxnet.dat` file. In the Constraint Manager enabled flow, information about electrical constraints on the schematic are stored in the `psdcmdb.dat` file and not in the `pstxnet.dat` file.

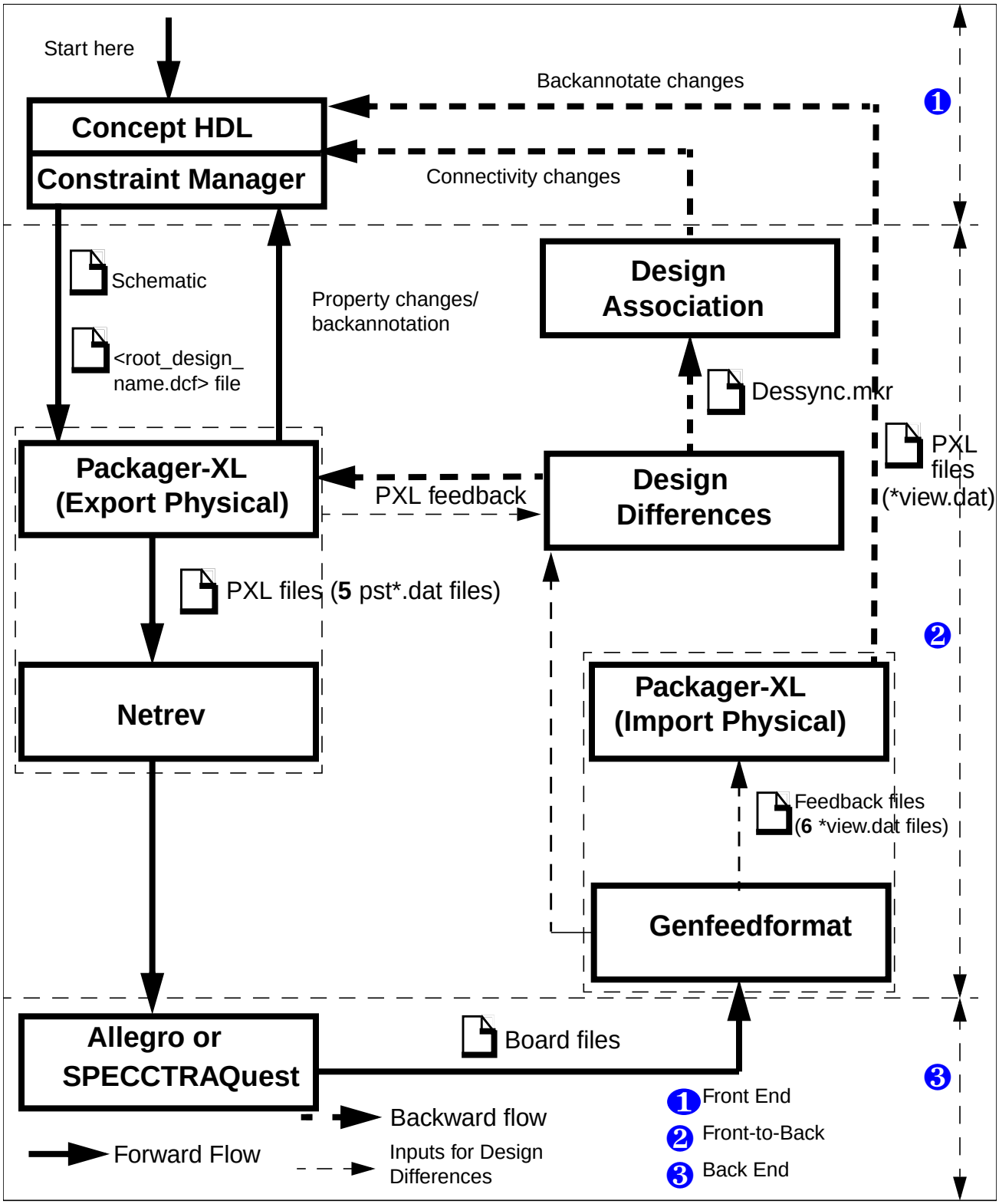
- **pstcmbc.dat**—Contains the electrical constraint information for the design used by the board. This information was generated during the last time when Import Physical was run on the board.

Note: If the <*root_design_name*>.dcf file is not present, Packager-XL runs in the traditional flow, where the information about electrical constraint properties are passed as normal properties in the `pstxnet.dat` file. For more information, see Traditional Flow on page 260.

The five `pst*.dat` files are used by `Netrev` to create or update the board. You can make changes in Allegro and then feed back the changes in the board to the schematic by running Import Physical (with *Generate Feedback Files* check box, *Package Design* check box and *Allegro* option selected). Import Physical allows you to overwrite all current electrical constraints in the schematic with the electrical constraint information in the Allegro board file or import only the electrical constraint information that has changed in the Allegro board file since the last import. Import Physical detects the presence of the <*root_design_name*>.dcf file and runs in the Constraint Manager enabled flow.

Note: If the <*root_design_name*>.dcf file exists in the `constraints` view of the root design, you are using the Constraint Manager enabled flow. The *Extract Constraints* check box in the *Import Physical* dialog box will be selected by default. You cannot deselect this check box because once you are in the Constraint Manager enabled flow, you cannot go back to the traditional flow.

Figure 11-2 Constraint Manager Enabled Flow



When you run Import Physical, `genfeedformat` creates the following six feedback files—`pinview.dat`, `netview.dat`, `funcview.dat`, `compview.dat`, `cmdbview.dat`, and `cmbcview.dat`. Note that the first four files are the same as those created in the traditional flow. The remaining two files contain electrical information as described below:

- **cmdbview.dat**—Describes the current electrical constraint information for the design. This file contains the latest electrical constraint information existing in the board.
- **cmbcview.dat**—Specifies the base copy of the electrical constraint information used by the Allegro board.

You can now use the feedback files to synchronize the schematic and the board by doing one of the following:

- Choose *Tools* > *Back Annotate* in Concept HDL to backannotate all the changes in the board to the schematic. Select the *Package Backannotation* check box in the *Backannotation* dialog box to backannotate all changes (excluding changes in electrical constraint information) in the board to the schematic. Select the *Constraint Backannotation* check box in the *Backannotation* dialog box to backannotate the last changes in electrical constraint information in the board or Constraint Manager to the schematic.
- Use the Design Differences (VDD) and Design Association (DA) tools to resolve individual connectivity and property differences between the schematic and the board.

  Use VDD to update the property differences either to the board or to the schematic. When you run VDD, it displays differences in properties between the schematic and the board in multiple windows. The differences in electrical constraints information in the schematic and the board are displayed in two windows—*Constraints Differences-Logical* and *Constraints Differences-Physical*.

  Note: In the traditional flow, differences in electrical constraints between the schematic and the board are displayed in the *Net Property Difference* window.

  Use DA to update the connectivity changes made in the board to the schematic. DA uses a file generated by VDD named `dessync.mkr` (which captures connectivity information) to guide you in updating the schematic.

For more information on the Constraint Manager enabled flow and the tools used in the flow, see the *Design Synchronization and Packaging User Guide* and the *Packager-XL Reference*.

Files Needed for Board Layout in Constraint Manager Enabled Flow

You may be creating the schematic yourself and getting the board layout done by a board designer. In such cases, after you package your design, you may be sending only the `pst*.dat` files to the board designer. If you are using the Constraint Manager enabled flow you must send the following five files to the board designer:

- pstchip.dat
- pstrxprt.dat
- pstxnet.dat
- pstcmdb.dat
- pstcmbc.dat

Moving from Traditional Flow to Constraint Manager Enabled Flow

You must perform the steps described in this section only:

- If you did not use Constraint Manager with Concept HDL 14.0 or 14.1 and do not use Constraint Manager with Concept HDL 14.2
- If you have performed the steps described in If you open a Concept HDL 13.6 design in Concept HDL 14.2 and do not want to use Constraint Manager with Concept HDL 14.2

If you are not using Constraint Manager with Concept HDL 14.2, you are using the traditional flow. For more information, see Traditional Flow on page 260. If you now want to use Constraint Manager with Concept HDL 14.2, you must do the following.

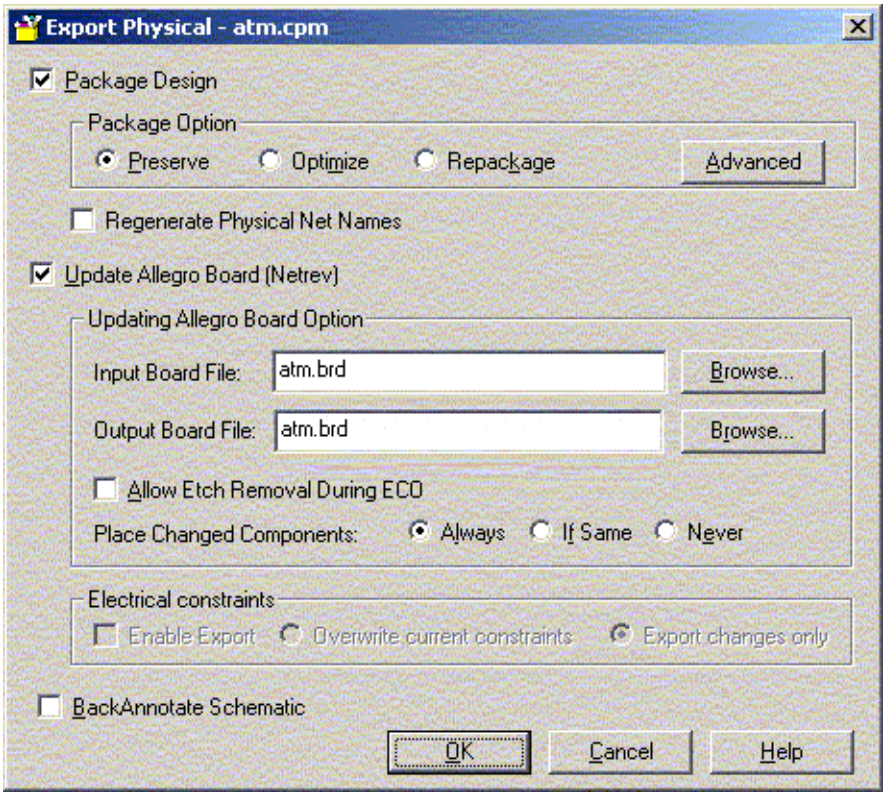
Note: Once you start using Constraint Manager with Concept HDL 14.2, you are in the Constraint Manager enabled flow. For more information, see Constraint Manager Enabled Flow on page 263.

Once you are in the Constraint Manager enabled flow, you cannot go back to the traditional flow.

1. Ensure that you are using Allegro and SPECCTRAQuest 14.2.

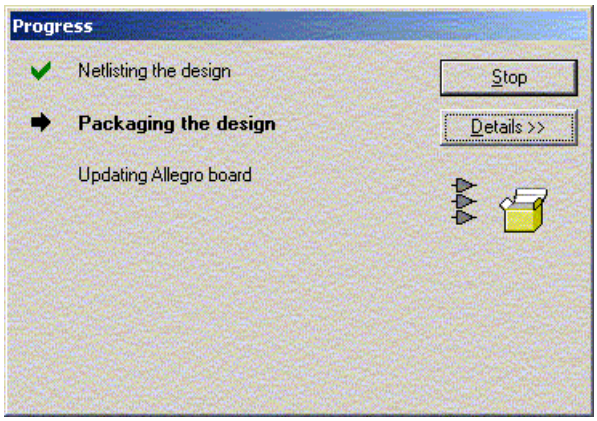
2. Open the design in Concept HDL 14.2.
3. Choose *File > Export Physical* in Concept HDL.

   The *Export Physical* dialog box appears.



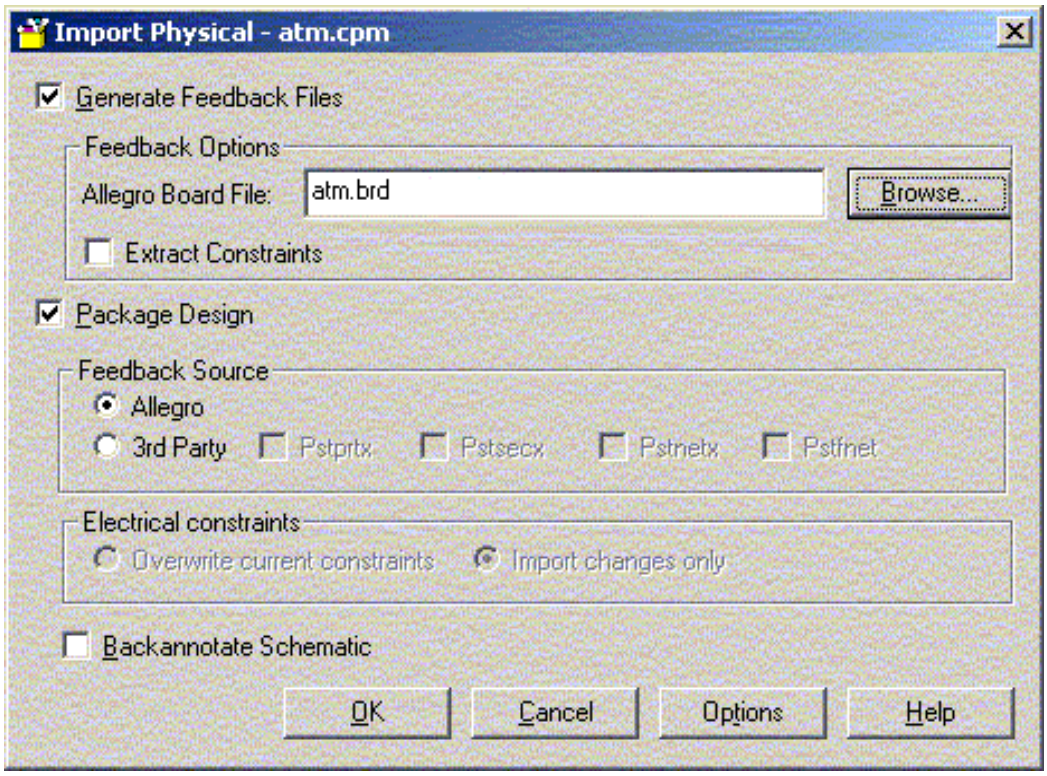
4. Select the *Package Design* and *Update Allegro Board (Netrev)* check boxes.
5. Click *OK*.

The *Progress* dialog box appears, displaying the progress of the Export Physical process.



6. Choose *File > Import Physical* in Concept HDL.

   The *Import Physical* dialog box appears.



7. Select the *Generate Feedback Files* check box.
8. Select the *Extract Constraints* check box.

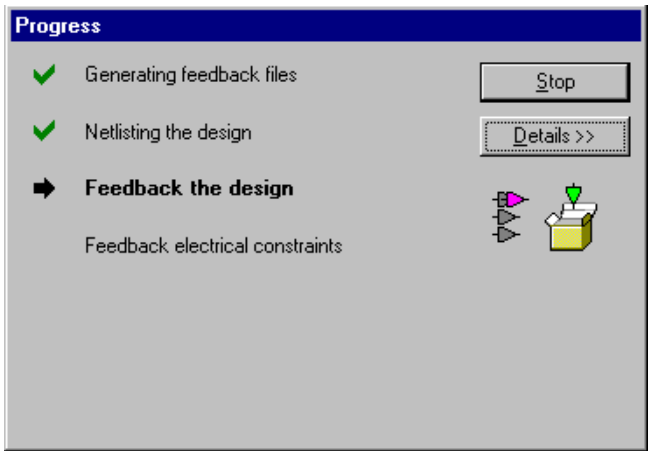
The following message box appears:

Design Sync
Once the Constraint Manager Flow is enabled, for this design, you cannot go back to the traditional flow. Continue extracting constraints ?
Yes
No

For more information on the traditional flow and the Constraint Manager enabled flow, see Traditional Flow on page 260 and Constraint Manager Enabled Flow on page 263.

9. Click *Yes*.
10. Select the *Package Design* check box and the *Allegro* option.
11. Click *OK*.

    The *Progress* dialog box appears, displaying the progress of the Import Physical process.



You can now start using Constraint Manager with Concept HDL 14.2.

Starting Constraint Manager

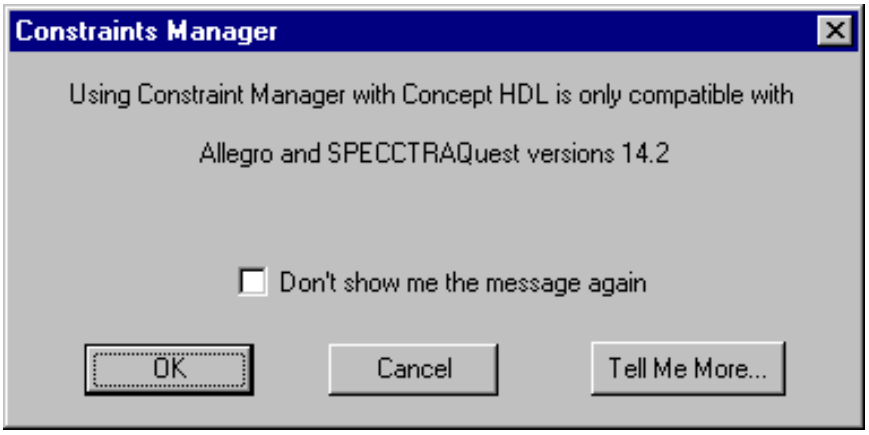
You can start Constraint Manager from Concept HDL only if you have the Concept HDL Expert and PCB Design Expert licenses.

To start Constraint Manager,

1. Choose *Tools* > *Constraints* > *Edit*.

If the *Attributes* dialog box is open, Concept HDL prompts you to save the changes. Click *Yes* to save the changes and to start Constraint Manager.

The following message box appears:


Constraints Manager
Using Constraint Manager with Concept HDL is only compatible with
Allegro and SPECCTRAQuest versions 14.2
Don't show me the message again
OK
Cancel
Tell Me More...

2. Click *OK* if you are using Allegro or SPECCTRAQuest 14.2.

For information on why Constraint Manager with Concept HDL is compatible only with Allegro or SPECCTRAQuest 14.2, see Why Constraint Manager with Concept HDL 14.2 is not compatible with Allegro or SPECCTRAQuest 14.1 or previous versions on page 272.

- If all the pages in the design are not saved, Concept HDL displays the list of schematic pages that have to be saved and prompts you to save and expand the design. Click *Yes* to save the pages, to expand the design, and to start Constraint Manager.
- If all the pages in the design are saved but the design is not in the Expanded or Occurrence Edit mode, Concept HDL prompts you to expand the design. Click *Yes* to expand the design and to start Constraint Manager.

The constraints on the schematic are pushed to Constraint Manager. You can view the latest constraints information in Constraint Manager.

The Constraint Manager titlebar displays `(connected to ConceptHDL)`. This indicates that Constraint Manager has been started from Concept HDL.

Important

If you want to use Constraint Manager with Concept HDL, you must start Constraint Manager only from Concept HDL. If you start Constraint Manager from Allegro, SPECCTRAQuest, or the UNIX or Windows command prompt, you cannot use Constraint Manager to manage constraints in Concept HDL.

Why Constraint Manager with Concept HDL 14.2 is not compatible with Allegro or SPECCTRAQuest 14.1 or previous versions

If you are using Constraint Manager with Concept HDL 14.2:

- You must use only Allegro or SPECCTRAQuest versions 14.2 with Concept HDL 14.2.
- You must not use Concept HDL 14.1 or a previous version with Allegro or SPECCTRAQuest versions 14.2

This is because, when you are using Constraint Manager with Concept HDL you will be using the new Constraint Manager enabled flow which is not supported by previous versions of Concept HDL, Allegro and SPECCTRAQuest. For more information, see Constraint Manager Enabled Flow on page 263.

Working with Electrical Constraints

The section describes the following topics:

- Capturing Electrical Constraints on page 272
- Displaying Electrical Constraints Captured in Constraint Manager on the Schematic on page 291
- Modifying Electrical Constraints on page 293
- Deleting Electrical Constraints on page 295
- Auditing Obsolete Objects on page 296

Capturing Electrical Constraints

You can capture electrical constraints using Concept HDL or Constraint Manager.

Cadence recommends that you capture constraints in Constraint Manager. Constraint Manager's user-friendly interface allows you to quickly capture and manage electrical constraints. Constraint Manager validates the constraint information that you enter and passes the information in the correct syntax to Concept HDL. If you add electrical constraint properties in Concept HDL, there is a possibility of error as the syntax of the value of some electrical constraint properties are complex. You may, however, have the occasional need to add an electrical constraint property directly in Concept HDL.

See the following topics for more information on capturing constraints:

- Using Constraint Manager to Capture Electrical Constraints on page 273

- Using Concept HDL to Add Electrical Constraint Properties on page 274
- Buses and pin-pairs are treated as special objects when working with electrical constraints. For more information on working with electrical constraints on buses and pin-pairs, see:
  - Working with Electrical Constraints on Buses on page 275
  - Working with Electrical Constraints on Pin-Pairs on page 282
- See Working with Electrical Constraints in Hierarchical Designs on page 287 for recommendations on working with electrical constraints in hierarchical designs.

Using Constraint Manager to Capture Electrical Constraints

In Constraint Manager, you can capture an electrical constraint on design objects in the following two ways:

- Create an electrical constraint set (ECSet) in Constraint Manager and assign the ECSet to a net in the *Net* worksheets.
- Specify the constraints directly on an object (net or pin-pair) using the *Net* worksheets.

  Note: Until the design is packaged, nets are displayed in Constraint Manager in the canonical (logical) format. After the design is packaged, nets are displayed in Constraint Manager using physical (packaged) names. For more information, see How Nets Are Displayed in Constraint Manager on page 314.

After you have completed capturing constraints, do the following:

1. Choose *File > Save* in Constraint Manager or Concept HDL.
2. Choose *Tools > Constraints > Update Schematic* in Concept HDL.

The constraints that you capture in Constraint Manager are updated in Concept HDL when you choose *Tools > Constraints > Update Schematic* in Concept HDL. You can view the constraints using the *Attributes* dialog box in Concept HDL. The constraints that you capture in Constraint Manager will not be automatically visible on the schematic in Concept HDL. To make a constraint visible in Concept HDL, add a placeholder for the constraint on the schematic. See Displaying Electrical Constraints Captured in Constraint Manager on the Schematic on page 291, for information on making constraints captured in Constraint Manager visible on the schematic in Concept HDL.

For more information on capturing constraints in Constraint Manager, see the documentation for *Constraint Manager*.

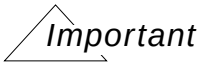
Tip

You can perform cross probing from Concept HDL to Constraint Manager to quickly locate nets on which you want to capture constraints in Constraint Manager. For more information, see Cross Probing between Concept HDL and Constraint Manager on page 312.

Using Concept HDL to Add Electrical Constraint Properties

Cadence recommends that you capture constraints in Constraint Manager and not in Concept HDL.

In Concept HDL, you can add electrical constraint properties on a net or pin on the schematic. See the *PCB Systems Properties Reference* for information on the electrical constraint properties that you can add on a net or pin in Concept HDL.



Important

You cannot add constraint properties in Concept HDL when Constraint Manager is running. Exit Constraint Manager if you want to add constraints in Concept HDL.

In Concept HDL, you can capture an electrical constraint on a net or pin in the following two ways:

1. Assign an existing electrical constraint set (ECSet) to a net in Concept HDL by adding the `ELECTRICAL_CONSTRAINT_SET` property on the net with its value being the name of the ECSet.

   For example, create an ECSet named `CONTROL` in Constraint Manager. If you want to assign the ECSet to a net named `CLOCK` in Concept HDL, add the `ELECTRICAL_CONSTRAINT_SET=CONTROL` property on the `CLOCK` net in Concept HDL.

   Note: You cannot define an ECSet in Concept HDL.

2. Add an electrical constraint property on a net or pin in Concept HDL.

   For example, if you want to specify the maximum crosstalk of a net named `CLOCK` as `0.5 mV`, add the `MAX_XTALK=.5` property on the `CLOCK` net in Concept HDL.

   Note the following when adding constraint properties in Concept HDL:

   - Cadence recommends that you specify the unit of value for a property. Otherwise, Constraint Manager assigns a default unit of value. For example, the default unit of value for the `MAX_XTALK` electrical constraint in Constraint Manager is `mV`.

- An electrical constraint property on a net in Concept HDL overrides the same constraint inherited from an ECSet assigned to the net.

  For example, suppose that you have assigned an ECSet that has the Max Xtalk (maximum crosstalk) defined as `0.6 mV` to a net. If you add the `MAX_XTALK=0.5 mV` electrical constraint property on the net, the `MAX_XTALK` electrical constraint property overrides the constraint defined in the ECSet. The maximum crosstalk for the net will now be `0.5 mV`.

  Cadence recommends that you minimize the number of such overrides. It becomes difficult to keep track of overrides when design requirements change.

- Complex constraints, such as constraints on pin-pairs, should be captured in Constraint Manager so that their syntax is validated. For more information, see Working with Electrical Constraints on Pin-Pairs on page 282.

- See Working with Electrical Constraints on Buses on page 275 for more information on capturing constraints on buses (vectored nets).

The constraints that you capture in Concept HDL are displayed in Constraint Manager when you start Constraint Manager from Concept HDL. In Constraint Manager, constraints are stored in cells in worksheets. For example, if you add the `MAX_XTALK` electrical constraint property in Concept HDL, the constraint is stored in the *Estimated XTalk* worksheet in Constraint Manager. For information on the cell in which a constraint is displayed, see Appendix B, “Property Mapping” of the *Constraint Manager Design Guide*.

Working with Electrical Constraints on Buses

If you capture a constraint on a bus (vectored net), the constraint is applicable to all the bits of the bus. You can also capture a constraint on a bit of a bus. A constraint captured on a bit of a bus overrides a constraint captured on the bus.

You can capture constraints on a bus or a bit of a bus in Constraint Manager or Concept HDL.

- For more information on capturing constraints on a bus or a bit of a bus in Constraint Manager, see the documentation for *Constraint Manager*. The constraints that you capture on a bus or bit in Constraint Manager will not be visible in Concept HDL unless you add placeholders for the constraints in the schematic. For more information, see Displaying Electrical Constraints Captured in Constraint Manager on the Schematic on page 291.

- See the following topics for more information on capturing constraints on buses in Concept HDL:

  - Capturing Electrical Constraints on a Bus in Concept HDL on page 276

- Capturing Electrical Constraints on a Bit of a Bus in Concept HDL on page 276
- Viewing Electrical Constraints on a Bit of a Bus in Concept HDL on page 280

Capturing Electrical Constraints on a Bus in Concept HDL

In Concept HDL, you can add an electrical constraint property on a bus. For more information, see Capturing Electrical Constraints on page 272.

Note: Cadence recommends that you capture constraints on a bus only in Constraint Manager because any constraint captured at the bus-level will be maintained only in Constraint Manager. Only bit-level constraints are synchronized between Concept HDL and Constraint Manager. If you add an electrical constraint property on a bus in Concept HDL, the constraint is pushed to Constraint Manager and displayed on the bits of the bus in Constraint Manager. If you later change the value of the electrical constraint property on the bus in Concept HDL, the new value will not be pushed to Constraint Manager. You can change the value of the constraint on the bus only in Constraint Manager.

Note: If you add an electrical constraint property on a bus in the Hierarchy or Expanded mode in Concept HDL and later change the value of the constraint on the bus in Constraint Manager, the new value will be visible on the schematic only in the Occurrence Edit mode in Concept HDL. You can also view the new value using the *Attributes* dialog box in the Expanded mode in Concept HDL.

For more information on the Hierarchy, Expanded, and Occurrence Edit modes in Concept HDL, see Modes in Concept HDL on page 113.

Capturing Electrical Constraints on a Bit of a Bus in Concept HDL

The following examples describe the procedure for capturing constraints on a bit of a bus in Concept HDL:

- Example 1: Capturing Constraints on a Bit of a Bus on page 276
- Example 2: Capturing Constraints on a Bit of a Bus on page 278
- Example 3: Capturing Constraints on a Bit Explicitly Tapped from a Bus on page 279

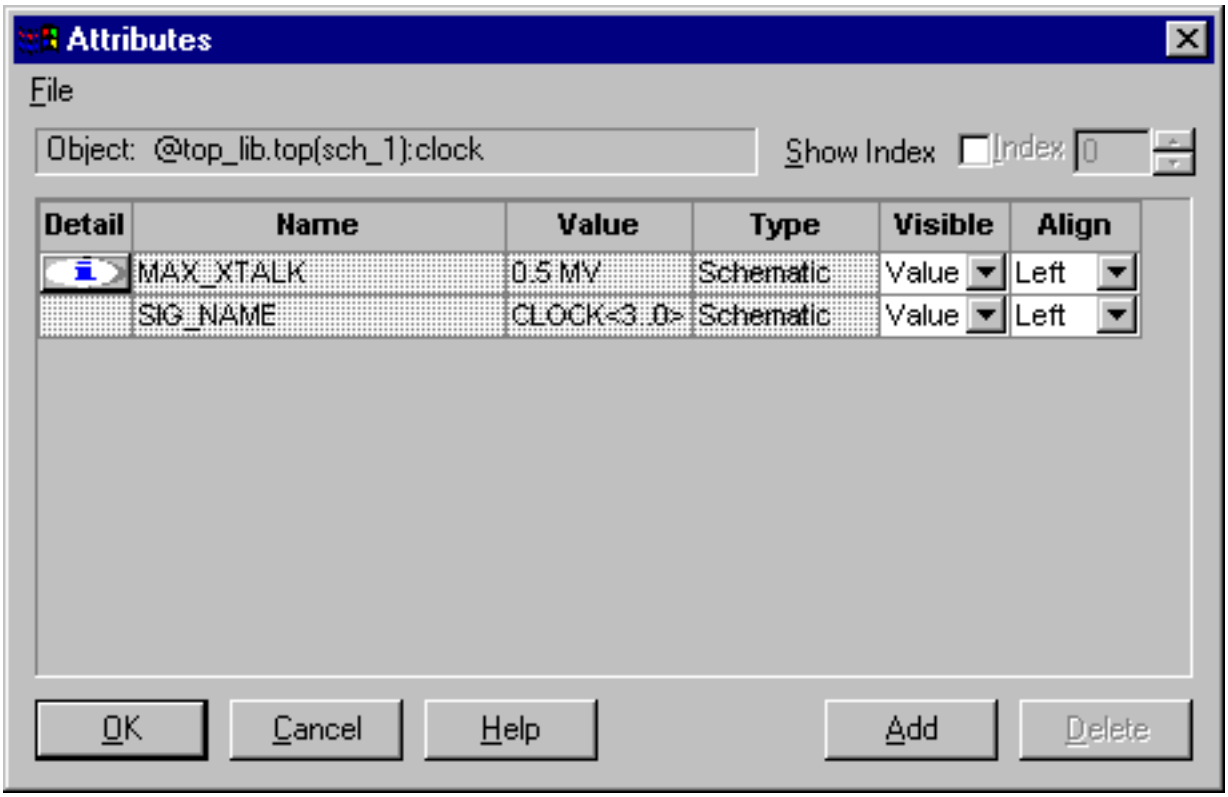
Example 1: Capturing Constraints on a Bit of a Bus

If you have a bus `CLOCK<3..0>` on the schematic, you can add electrical constraint properties on the bits of the bus only if you are in the Occurrence Edit mode in Concept HDL.

Suppose that you have a constraint to define the maximum crosstalk of the bus `CLOCK<3..0>` as `0.5 mV`. To specify the maximum crosstalk of the 3rd bit of the bus `CLOCK<3..0>` as `0.4 mV`, and the maximum tolerated voltage overshoot for the 3rd bit as `50 mV`, do the following:

1. Exit Constraint Manager if you have started it from Concept HDL.
2. In Concept HDL, choose *Tools* > *Occurrence Edit*.
3. Choose *Text* > *Attributes* in Concept HDL.
4. Click the `CLOCK<3..0>` bus.

   The *Attributes* dialog box appears.



5. Select the *Show Index* check box.

6. Type 3 in the *Index* field.

   The properties for the 3rd bit of the bus are displayed. The MAX_XTALK property has the value 0.5 mV because the MAX_XTALK constraint captured on the bus CLOCK<3..0> applies to all bits of the bus.

7. In the *Value* field next to the MAX_XTALK property, modify the value from 0.5 mV to 0.4 mV.

8. Click *Add*.

9. Type MAX_OVERSHOOT in the *Name* field.

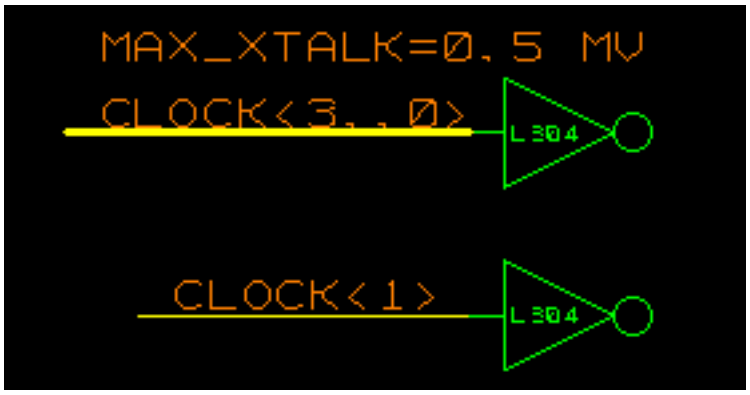
10. Type 50 mV in the *Value* field next to the MAX_OVERSHOOT property to specify the maximum tolerated voltage overshoot of the bit as 50 mV.

11. Click *OK* to save the changes and to close the *Attributes* dialog box.

Example 2: Capturing Constraints on a Bit of a Bus

If you have a bus CLOCK<3..0> and a bit CLOCK<1> on the schematic, you can add electrical constraint properties on the bit CLOCK<1> in the Hierarchy, Expanded, and Occurrence Edit modes in Concept HDL.

Note: If you add an electrical constraint property on the bit CLOCK<1> in the Hierarchy or Expanded mode in Concept HDL and later change the value of the constraint on the bit in Constraint Manager, the new value will be visible only in the Occurrence Edit mode in Concept HDL. You can also view the new value using the *Attributes* dialog box in the Expanded mode in Concept HDL.



Suppose that you have a constraint to define the maximum crosstalk of the bus CLOCK<3..0> as 0.5 mV. To specify the maximum crosstalk of the 1st bit of the bus as 0.4 mV, and the maximum tolerated voltage overshoot for the 1st bit as 50 mV, do the following:

1. Exit Constraint Manager if you have started it from Concept HDL.
2. Choose *Text* > *Attributes* in Concept HDL.
3. Click on the bit `CLOCK<1>`.

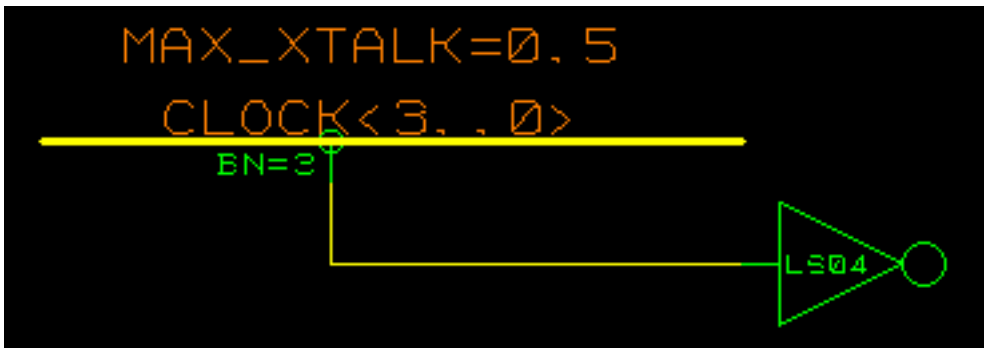
   The *Attributes* dialog box appears. The `MAX_XTALK` property has the value `0.5 mV` because the `MAX_XTALK` constraint captured on the bus `CLOCK<3..0>` applies to all the bits of the bus.
4. In the *Value* field next to the `MAX_XTALK` property, modify the value from `0.5 mV` to `0.4 mV`.
5. Click *Add*.
6. Type `MAX_OVERSHOOT` in the *Name* field.

   Type `50 mV` in the *Value* field next to the `MAX_OVERSHOOT` property to specify the maximum tolerated voltage overshoot of the bit as `50 mV`.
7. Click *OK* to save the changes and close the *Attributes* dialog box.

Example 3: Capturing Constraints on a Bit Explicitly Tapped from a Bus

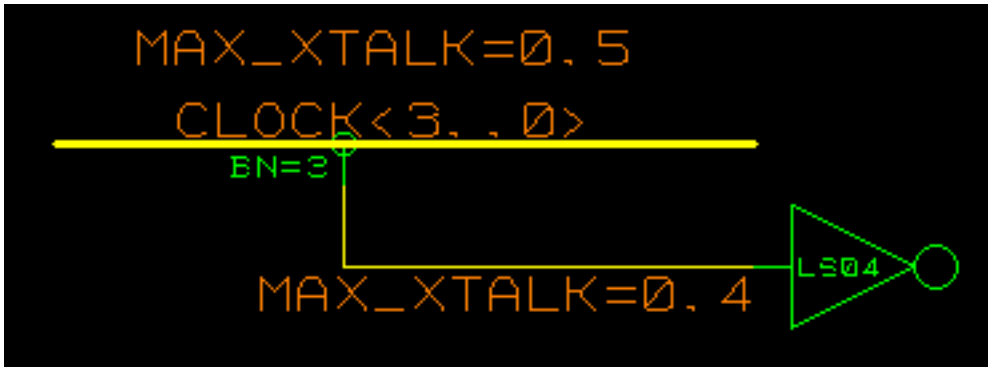
Suppose that you have captured a constraint to define the maximum crosstalk of a bus `CLOCK<3..0>` as `0.5 mV`. If you have explicitly tapped the 3rd bit of the bus `CLOCK<3..0>` as shown in the following circuit, you can define the maximum crosstalk of the third bit as `0.4 mV` in the Hierarchy, Expanded, and Occurrence Edit modes in Concept HDL, as shown below:

Note: If you add an electrical constraint property on the tapped bit in the Hierarchy or Expanded mode in Concept HDL and later change the value of the constraint on the tapped bit in Constraint Manager, the new value will be visible only in the Occurrence Edit mode in Concept HDL. You can also view the new value using the *Attributes* dialog box in the Expanded mode in Concept HDL.



1. Exit Constraint Manager if you have started it from Concept HDL.
2. Choose *Text* > *Attributes* in Concept HDL.
3. Click on the tapped bit.

   The *Attributes* dialog box appears. The MAX_XTALK property has the value `0.5 mV` because the MAX_XTALK constraint captured on the bus `CLOCK<3..0>` applies to all the bits of the bus.
4. In the *Value* field next to the MAX_XTALK property, modify the value from `0.5 mV` to `0.4 mV`.
5. Click *OK* to save the changes and close the *Attributes* dialog box.



Viewing Electrical Constraints on a Bit of a Bus in Concept HDL

The following examples explain how you can view constraints on a bit of a bus in Concept HDL:

- Example 1: Viewing Constraints on a Bit of a Bus on page 280
- Example 2: Viewing Constraints on a Bit of a Bus on page 281
- Example 3: Viewing Constraints on a Bit Explicitly Tapped from a Bus on page 282

Example 1: Viewing Constraints on a Bit of a Bus

If you have a bus `CLOCK<3..0>` on the schematic, the constraints on the bits of the bus will not be displayed on the schematic. You must change to the Occurrence Edit mode in Concept HDL and use the *Attributes* dialog box to view the constraints on the bits of the bus.

To view the constraints on the 3rd bit of the bus CLOCK<3..0> in Concept HDL, do the following:

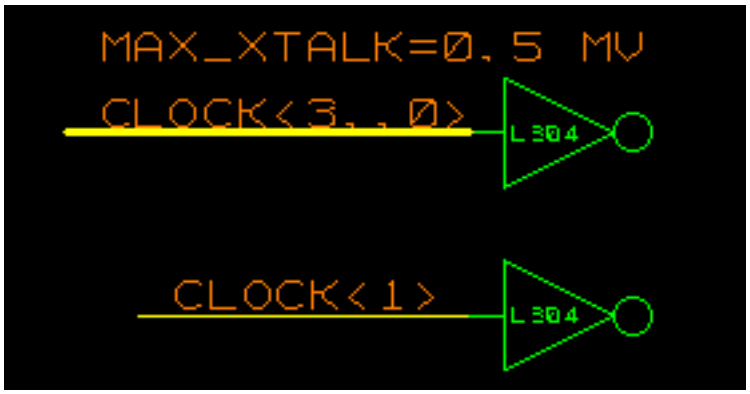
1. In Concept HDL, choose *Tools* > *Occurrence Edit*.
2. Choose *Text* > *Attributes*.
3. Click the CLOCK<3..0> bus in Concept HDL.

   The *Attributes* dialog box appears.
4. Select the *Show Index* check box.
5. Type 3 in the *Index* field.

   The properties for the 3rd bit of the bus are displayed.

Example 2: Viewing Constraints on a Bit of a Bus

Suppose that you have a bus CLOCK<3..0> and a bit CLOCK<1> on the schematic. The constraints on the bit CLOCK<1> will not be displayed on the schematic. You must change to the Expanded or Occurrence Edit mode in Concept HDL and use the *Attributes* dialog box to view the constraints on the bit CLOCK<1>.



To view the constraints on the 1st bit of the bus CLOCK<3..0> in Concept HDL, do the following:

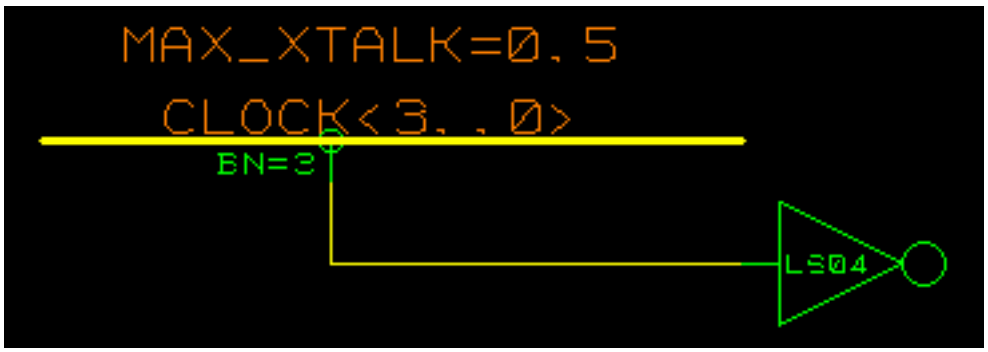
1. Choose *Tools* > *Expand* or *Tools* > *Occurrence Edit*.

2. Choose *Text* > *Attributes*.
3. Click on the `CLOCK<1>` bit in Concept HDL.

   The *Attributes* dialog box appears displays the constraints on the bit.

Example 3: Viewing Constraints on a Bit Explicitly Tapped from a Bus

Suppose that you have explicitly tapped the 3rd bit of a bus `CLOCK<3..0>` as shown in the following circuit. The constraints on the 3rd bit of the bus `CLOCK<3..0>` will not be displayed on the schematic. You must change to the Expanded or Occurrence Edit mode in Concept HDL and use the *Attributes* dialog box to view the constraints on the 3rd bit.



To view the constraints on the 3rd bit that is explicitly tapped from the bus `CLOCK<3..0>` in Concept HDL, do the following:

1. Choose *Tools* > *Expand* or *Tools* > *Occurrence Edit*.
2. Choose *Text* > *Attributes*.
3. Click on the tapped bit in Concept HDL.

   The *Attributes* dialog box appears displays the constraints on the bit.

Working with Electrical Constraints on Pin-Pairs

A pin-pair represents a pair of logically connected pins that form a driver-receiver connection. Pin-pairs may not be directly connected but they must exist on the same net. You use pin-pairs to capture specific pin-to-pin constraints for a net.

Note: Cadence recommends that you capture constraints on pin-pairs only in Constraint Manager and not in Concept HDL.

Capturing Constraints on Pin-Pairs in Constraint Manager

Before you capture constraints on a pin-pair in Constraint Manager, ensure that the components that form the pin-pair have reference designators (LOCATION property) and pin numbers (PN property). To assign reference designators and pin numbers for the components that form the pin-pair, do one of the following:

- Package your design. When you package your design, reference designators and pin numbers are automatically assigned on the components in the design. For information on packaging the design, see the *Design Synchronization and Packaging User Guide*.
- Section the components that form the pin-pair and add the `LOCATION=<value>` property on the components.

  Note: When you section a component, pin numbers are automatically assigned on the component.

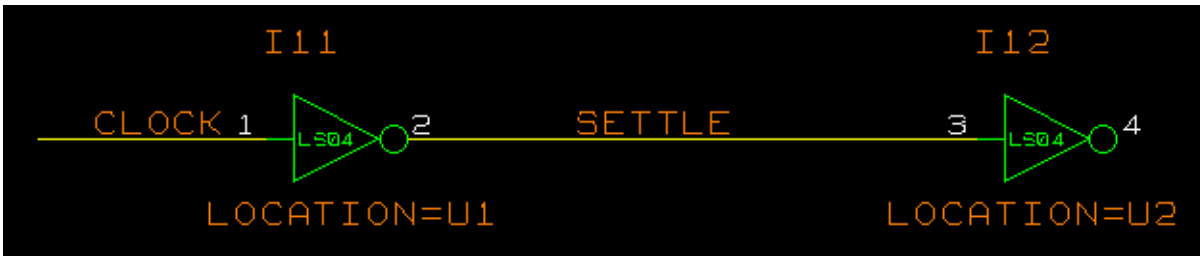
The procedure for sectioning the components that form the pin-pair, adding the `LOCATION=<value>` property on the components and then capturing the constraint on the pin-pair in Constraint Manager is explained below using an example circuit.

To capture the minimum first switch delay for the pin-pair, do the following:

1. Choose *Component* > *Section* and section both the `ls04` components as below:
   - Section instance `I11` of the `ls04` component to have pin numbers 1 and 2.
   - Section instance `I12` of the `ls04` component to have pin numbers 3 and 4.

   Pin 1 of instance `I11` of the `ls04` component and Pin 3 of instance `I12` of the ls04 component make the driver/receiver pin-pair.
2. Add the `LOCATION` property, say `LOCATION=U1` property on instance `I11` of the `lso4` component.

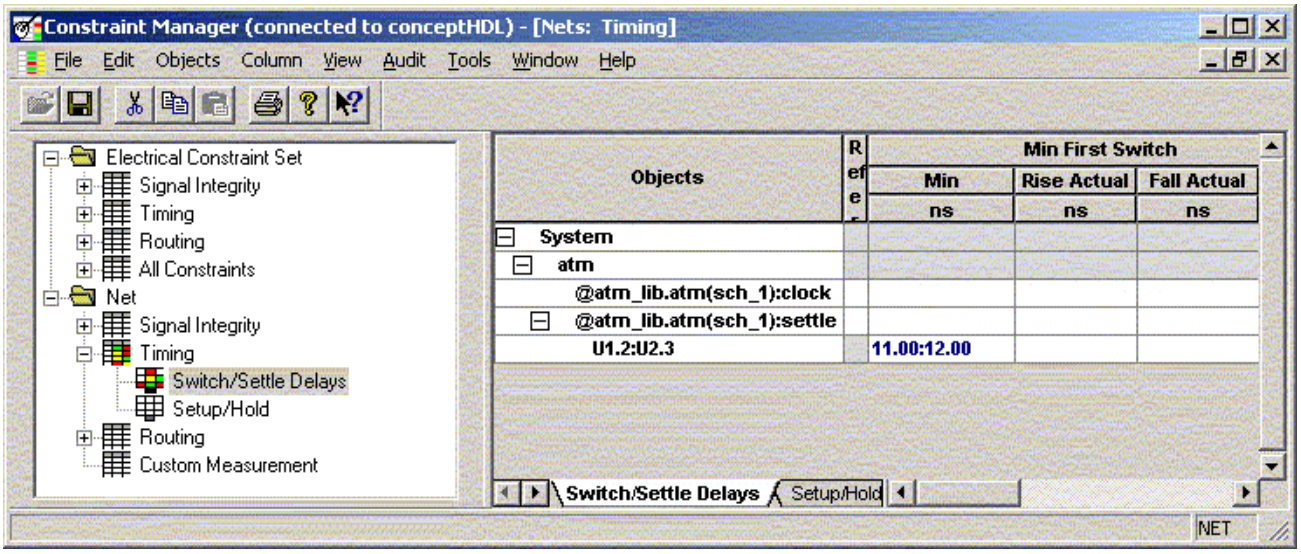
3. Add the `LOCATION` property, say `LOCATION=U2` property on instance `I12` of the `lso4` component.



4. Choose *File > Save* in Concept HDL.
5. Open the *Switch/Settle Delays* worksheet in Constraint Manager.
6. Create the pin pair `U1.2:U2.3` for the net `SETTLE` in Constraint Manager.

   For information on creating pin-pairs in Constraint Manager, see the documentation for *Constraint Manager*.

7. To capture the minimum first switch delay constraint with rising edge as `11 ns` and falling edge as `12 ns`, type `11:12` in the *Min* column next to the `U1.2:U2.3` pin pair.



For more information on capturing constraints on pin-pairs in Constraint Manager, see the documentation for *Constraint Manager*.

Capturing Constraints on Pin-Pairs in Concept HDL

You can capture constraints on pin-pairs in Concept HDL before or after packaging your design.

Note: Cadence recommends that you capture constraints on pin-pairs only in Constraint Manager and not in Concept HDL.

The procedure for capturing constraints on pin-pairs in Concept HDL is described below:

- Capturing Electrical Constraints on a Pin-Pair in Concept HDL before Packaging Your Design on page 285
- Capturing Electrical Constraints on a Pin-Pair in Concept HDL after Packaging Your Design on page 286

Capturing Electrical Constraints on a Pin-Pair in Concept HDL before Packaging Your Design

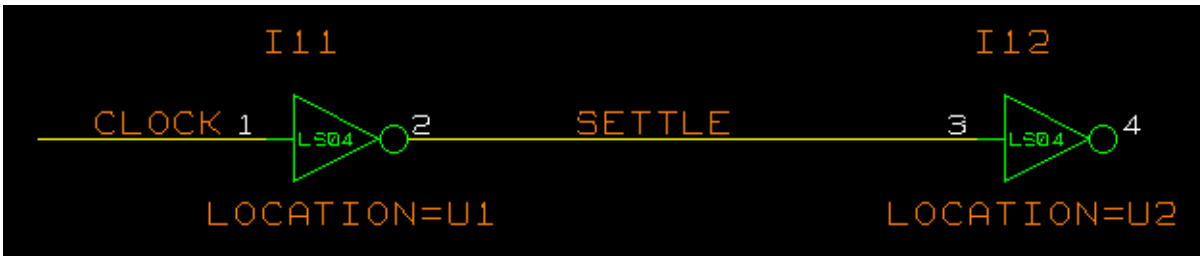
The procedure for capturing constraints on a pin-pair in Concept HDL before packaging the design is explained below using a sample circuit.

To capture the minimum first switch delay for the pin-pair, do the following:

1. Exit Constraint Manager if you have started it from Concept HDL.
2. Choose *Component* > *Section* and section both the `ls04` components as mentioned below:
   - Section instance `I11` of the `ls04` component so that it has pin numbers 1 and 2
   - Section instance `I12` of the `ls04` component so that it has pin numbers 3 and 4

   Pin 1 of instance `I11` of the `ls04` component and Pin 3 of instance `I12` of the `ls04` component form the driver/receiver pin-pair.
3. Add the `LOCATION` property, say `LOCATION=U1` property, on instance `I11` of the `ls04` component.

4. Add the `LOCATION` property, say `LOCATION=U2` property, on instance `I12` of the `ls04` component.



5. Add the constraint property `MIN_FIRST_SWITCH=U1.2:U2.3:11 ns:12 ns` on the net `SETTLE` to capture the minimum first switch delay (with rising edge as `11 ns` and falling edge as `12 ns`) for the pin-pair.

   For more information on the `MIN_FIRST_SWITCH` property, see the *PCB Systems Properties Reference*.

6. Choose *Tools* > *Constraints* > *Edit* to start Constraint Manager.

   The constraint on the pin-pair is displayed in Constraint Manager.

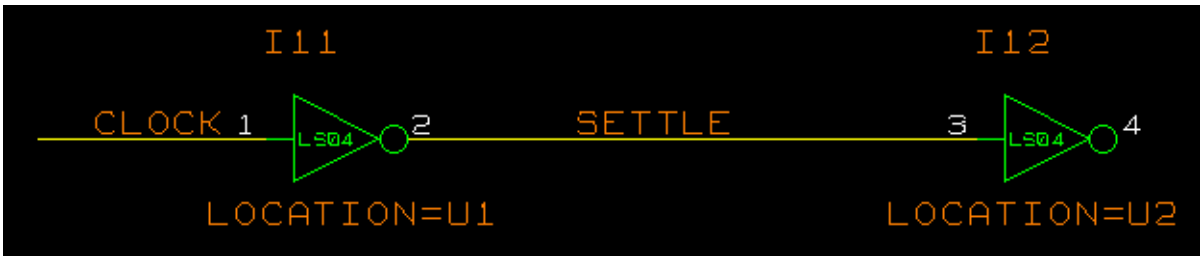
Capturing Electrical Constraints on a Pin-Pair in Concept HDL after Packaging Your Design

The procedure for capturing constraints on a pin-pair in Concept HDL after packaging the design is explained below using a sample circuit.

To capture the minimum first switch delay for the pin-pair, do the following:

1. Exit Constraint Manager if you have started it from Concept HDL.
2. Choose *Tools* > *Constraints* > *Update Schematic*.

3. Package the design. For information on packaging the design, see the *Design Synchronization and Packaging User Guide*.



4. Add the constraint property `MIN_FIRST_SWITCH=U1.2:U2.3:11 ns:12 ns` on the net `SETTLE` to capture the minimum first switch delay (with rising edge as `11 ns` and falling edge as `12 ns`) for the pin-pair, where:

   - `U1` is the value of the `LOCATION` property on the first instance of `ls04`,
   - 2 is the value of the `PN` property on the driver pin,
   - `U2` is the value of the `LOCATION` property on the second instance of `ls04` and
   - 3 is the value of the `PN` property on the receiver pin

   For more information on the `MIN_FIRST_SWITCH` property, see the *PCB Systems Properties Reference*.

5. Choose *Tools* > *Constraints* > *Edit* to start Constraint Manager.

   The constraint on the pin-pair is displayed in Constraint Manager.

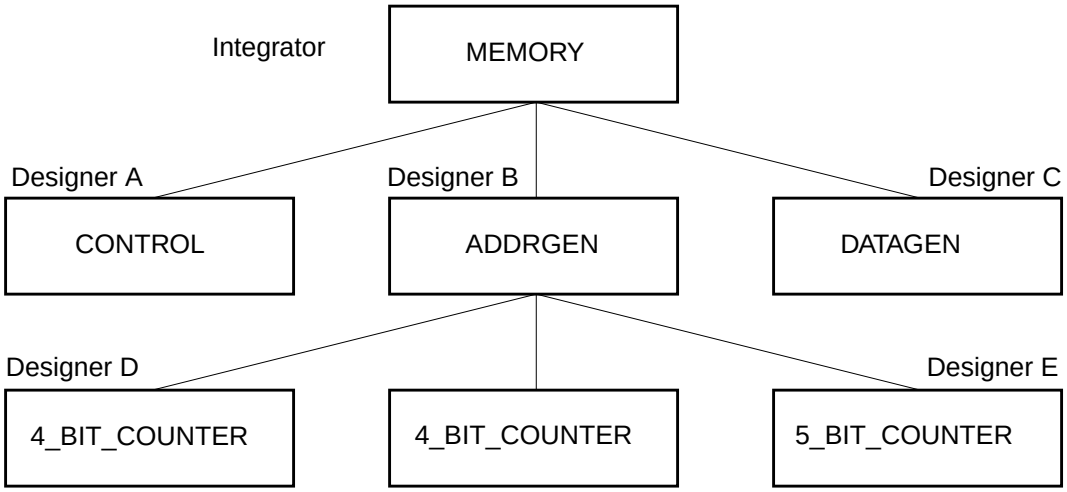
Working with Electrical Constraints in Hierarchical Designs

You need to keep the following things in mind when working with constraints in hierarchical designs:

- Team Design on page 288
- Constraints on a Block Instantiated Once in a Hierarchical Design on page 289
- Constraints on a Block Instantiated More than Once in a Design on page 290
- Constraints on Read-Only Blocks on page 290

Team Design

Teams of designers may be working on a hierarchical design. In this team design environment, each designer may be working on a block in the hierarchical design. After all the designers working on the blocks have completed their work, the Integrator (a Team Leader or a designated person) integrates all the blocks into the top level design.



In the above example of a hierarchical design `MEMORY`, teams of designers work on different blocks of the design. After all the designers working on the blocks have completed their work, the Integrator (a Team Leader or a designated person) integrates all the blocks into the top level design (`MEMORY`).

Cadence recommends the following for managing constraints in a team design environment:

- The designers working on the blocks must not use Constraint Manager to manage constraints in the lower level blocks. They must add electrical constraint properties in the schematic for the lower level blocks.

  For example, Designer B must not use Constraint Manager to manage constraints in the schematic for the `ADDRGEN` block. He must add electrical constraint properties on the schematic for the block.

- A block is instantiated more than once in a design is called a reuse or replicated block. For example, the `4_BIT_COUNTER` is a replicated block because it is instantiated twice in the schematic for the `ADDRGEN` block.

  The designer working on a block must not add electrical constraint properties on a replicated block in the Occurrence Edit mode in Concept HDL. This is because such occurrence properties will not be pushed to Constraint Manager. Cadence recommends that after the Integrator integrates all the lower level blocks into the top-level design, he

should add constraint properties on replicated blocks in the Occurrence Edit mode in Concept HDL. For more information on the Occurrence Edit mode in Concept HDL, see Occurrence Edit Mode on page 114.

For example, the block `4_BIT_COUNTER` is instantiated twice in the schematic for the `ADDRGEN` block. Designer B must not add electrical constraint properties on the `4_BIT_COUNTER` block in the Occurrence Edit mode in Concept HDL. After the Integrator has integrated all the lower level blocks into the top-level design `MEMORY`, he should add occurrence properties on the replicated block `4_BIT_COUNTER` in the schematic for the block `ADDRGEN`.

- When the Integrator starts Constraint Manager from Concept HDL after integrating all the lower level blocks into the top-level design `MEMORY`, the electrical constraint properties added in the lower level blocks (including the constraint properties added by the Integrator in the Occurrence Edit mode on replicated blocks in the schematic for lower level blocks) are displayed in Constraint Manager.

Constraints on a Block Instantiated Once in a Hierarchical Design

- If you have instantiated a block only once in a design and choose *Tools > Constraints > Update Schematic* or run backannotation, any changes to constraints in the block will be displayed in the Hierarchy, Expanded, and Occurrence Edit modes. This is because properties on the schematic for a block instantiated only once in a design can be backannotated to the schematic for the block.

- Add an electrical constraint property on a net in the schematic for a block and delete the constraint in Constraint Manager. The following message is displayed in the `concept2cm.log` file when you start Constraint Manager from Concept HDL:

  ```
  Property <property_name> has been marked for deletion on net
  <canonical_name_for_net> and will not be pushed to CMDB. Please do
  backannotation to delete EC from the schematic.
  ```

  The above message indicates that the constraint property on the net has been deleted in Constraint Manager but exists on the net in Concept HDL. The constraint property on the net in Concept HDL will not be pushed to Constraint Manager. Choose *Tools > Constraints > Update Schematic* to delete the constraint on the net in Concept HDL.

  Note: If you do not choose *Tools > Constraints > Update Schematic* in Concept HDL but exit Constraint Manager, modify the value of the constraint property on the net in Concept HDL and choose *File > Save* in Concept HDL, the constraint with the new value will be visible in Constraint Manager.

Constraints on a Block Instantiated More than Once in a Design

- If you have instantiated a block more than once in a design and choose *Tools > Constraints > Update Schematic* or run backannotation, any changes to constraints in the blocks will be visible only in the Occurrence Edit mode in Concept HDL. This is because properties on the schematic for blocks instantiated more than once in a design cannot be backannotated to the schematic for the block.
- Add an electrical constraint property on a net in the schematic for a block that is instantiated more than once in a design and delete the constraint in Constraint Manager. The following message is displayed in the `concept2cm.log` file when you start Constraint Manager from Concept HDL:

  ```
  Schematic property <property_name> lying in a reuse block has been marked for
  deletion on net <canonical_name_for_net> and will not be pushed to CMDB.
  ```

  The above message indicates that the constraint property on the net has been deleted in Constraint Manager but exists on the net in Concept HDL. The constraint property on the net in Concept HDL will not be pushed to Constraint Manager.

  Note: If you choose *Tools > Constraints > Update Schematic*, the constraint on the net will not deleted because it is a replicated block.

  If you exit Constraint Manager, modify the value of the constraint property on the net in Concept HDL and choose *File > Save* in Concept HDL, the constraint with the new value will be visible in Constraint Manager.

Constraints on Read-Only Blocks

If an electrical constraint property or a placeholder for a constraint property is present on a read-only block used in a hierarchical design and you modify or delete the constraint in Constraint Manager, errors will be displayed when you backannotate the schematic. This is because you do not have write permissions in the read-only block. To avoid this, add the `NO_BACKANNOTATE=ALL` property on the read-only block in the schematic.

Note: The `NO_BACKANNOTATE=ALL` property applies only to the block on which it is added and not to its child blocks.

For more information on using read-only blocks in your design, see Using Read-only Blocks in Your Design on page 324.

Displaying Electrical Constraints Captured in Constraint Manager on the Schematic

The constraints that you capture in Constraint Manager are updated in Concept HDL when you choose *Tools* > *Constraints* > *Update Schematic* in Concept HDL. You can view the constraints using the *Attributes* dialog box in the Expanded or Occurrence Edit mode in Concept HDL. The constraints that you capture in Constraint Manager will not be automatically visible on the schematic in Concept HDL. To make such constraints visible in Concept HDL, add placeholders for the constraints on the schematic.

In Constraint Manager, constraints are stored in cells in worksheets. Each constraint captured in Constraint Manager is represented by a corresponding electrical constraint property in Concept HDL. You can make a constraint captured in Constraint Manager visible in Concept HDL by adding a placeholder for the corresponding electrical constraint property in Concept HDL. For information on the cell in which a constraint is displayed in Constraint Manager and its corresponding electrical constraint property in Concept HDL, see Appendix B, “Property Mapping” of the *Constraint Manager Design Guide*.

Note: You do not have to exit Constraint Manager if you want to add placeholders for constraints on the schematic.

For example, if you have specified the maximum crosstalk of a net `CLOCK` as `0.5 mV` in the *Estimated Xtalk* worksheet in Constraint Manager, the constraint will not be visible on the net `CLOCK` in Concept HDL. To display the maximum crosstalk constraint on the net `CLOCK` in Concept HDL, add a placeholder for the `MAX_XTALK` constraint property on the net by doing the following:

1. Switch to the Expanded or Occurrence Edit mode in Concept HDL.
2. Choose *Text* > *Attributes* in Concept HDL.
3. Click the net `CLOCK`.

   The *Attributes* dialog box appears. You can see the `MAX_XTALK` property with visibility set to *None* in the *Attributes* dialog box.
4. In the *Visible* drop-down list next to the `MAX_XTALK` property, select the visibility of the `MAX_XTALK` property as *Name*, *Value*, or *Both*.
5. Click *OK* to save the changes and close the *Attributes* dialog box.
6. Choose *File* > *Save* in Concept HDL.
7. Choose *Tools* > *Constraints* > *Update Schematic*.

   If you had set the visibility of the `MAX_XTALK` constraint property to *Both* in step 4, the property `MAX_XTALK=0.5000:0.5000` is displayed on the net `CLOCK` on the schematic

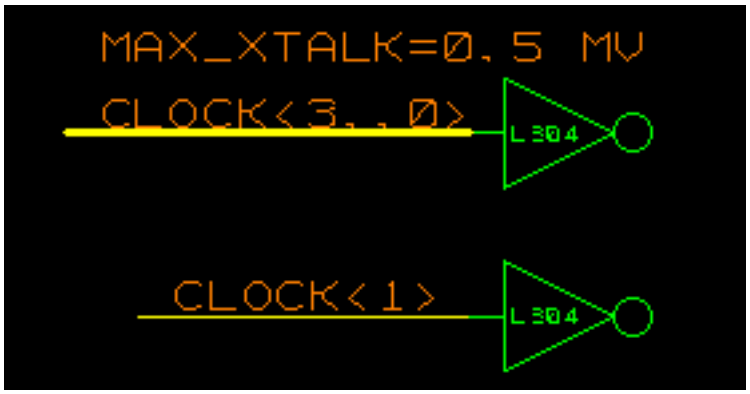
when you are in the Hierarchy, Expanded, or Occurrence Edit mode in Concept HDL. This property serves as a placeholder for the maximum crosstalk constraint captured in Constraint Manager. The placeholder is a coordinate on the schematic where the property will be displayed. If you modify the value of the constraint in Constraint Manager, the value for the placeholder is updated on the schematic when you choose *Tools > Constraints > Update Schematic* in Concept HDL. If you delete the constraint in Constraint Manager, the placeholder for the constraint is deleted when you choose *Tools > Constraints > Update Schematic* in Concept HDL.

Tip

You can perform cross probing from Constraint Manager to Concept HDL to quickly locate nets on which you want to add placeholders for constraints in Concept HDL. For more information, see Cross Probing between Concept HDL and Constraint Manager on page 312.

Note the following when adding placeholders for constraints in Concept HDL:

- You can add placeholders for constraints on nets and pins in Concept HDL.
- The placeholder for a constraint on a bus will display the constraint value only in the Occurrence Edit mode in Concept HDL. The placeholder for a constraint on a bus will not display the constraint value in the Hierarchy and Expanded mode in Concept HDL because constraints on a bus cannot be backannotated to the schematic.
- The placeholder for a constraint on a bit of a bus will display the constraint value only in the Occurrence Edit mode in Concept HDL. The placeholder for a constraint on a bit of a bus will not display the constraint value in the Hierarchy and Expanded mode in Concept HDL because constraints on a bit cannot be backannotated to the schematic.
  - If you have a bus and a bit on the schematic as shown in the following figure, you can add placeholders for constraints on the bit.



- If you have explicitly tapped a bit of a bus, you can add placeholders for constraints on the tapped bit.

Modifying Electrical Constraints

This section describes the procedures for modifying electrical constraints in Constraint Manager and in Concept HDL.

Note: Cadence recommends that you modify constraints in Constraint Manager and not in Concept HDL.

See the following topics for more information on modifying constraints:

- Modifying Electrical Constraints in Constraint Manager on page 293
- Modifying Electrical Constraints in Concept HDL on page 295
- Buses and pin-pairs are treated as special objects when working with electrical constraints. For more information on working with electrical constraints on buses and pin-pairs, see:
  - Working with Electrical Constraints on Buses on page 275
  - Working with Electrical Constraints on Pin-Pairs on page 282
- See Working with Electrical Constraints in Hierarchical Designs on page 287 for recommendations on working with electrical constraints in hierarchical designs.

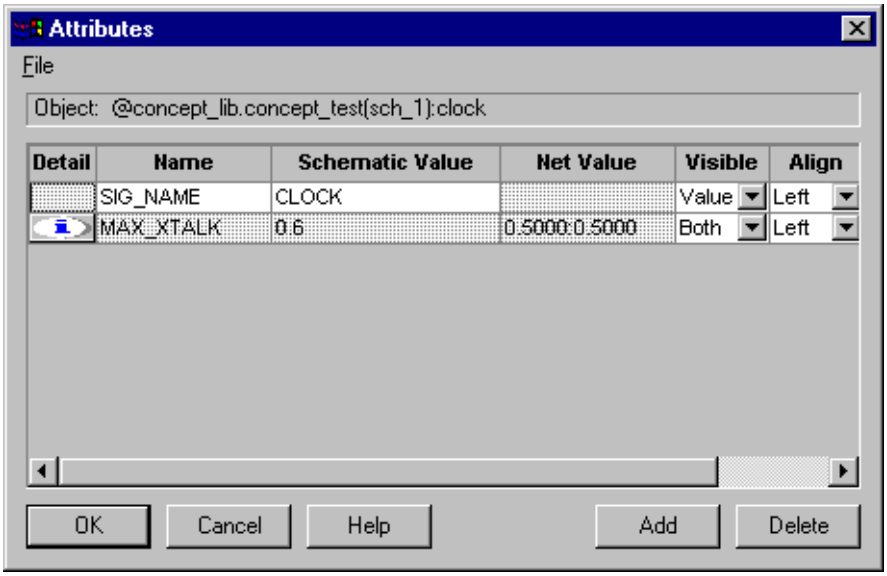
Modifying Electrical Constraints in Constraint Manager

If you have captured an electrical constraint in Constraint Manager or added an electrical constraint property in Concept HDL, you can modify the constraint in Constraint Manager.

To modify constraints in Constraint Manager, do the following:

1. Start Constraint Manager from Concept HDL.
2. Modify the constraint in Constraint Manager.
3. Choose *File > Save* in Constraint Manager or Concept HDL.

   If there is a constraint property on the schematic in Concept HDL, the changes will be visible in Concept HDL only if you are in the Occurrence Edit mode. If you are in the Expanded mode in Concept HDL, you can view the changes in the *Attributes* dialog box.



   In the above figure, the *Schematic Value* field next to the `MAX_XTALK` electrical constraint property displays the old value for the constraint and the *Net Value* field displays the modified value for the constraint on the net `CLOCK`.

4. Choose *Tools > Constraints > Update Schematic* in Concept HDL.

   If there is a constraint property or a placeholder for the constraint property in the schematic, the changes will be visible in the Hierarchy and Expanded modes in Concept HDL.

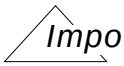
Note: If you had added the electrical constraint property in the Occurrence Edit mode in Concept HDL, the change will be visible only in the Occurrence Edit mode.

For more information on modifying constraints in Constraint Manager, see the documentation for *Constraint Manager*.

Modifying Electrical Constraints in Concept HDL

Cadence recommends that you modify constraints in Constraint Manager and not in Concept HDL.

You cannot modify constraint properties in Concept HDL when Constraint Manager is running. Exit Constraint Manager if you want to modify the electrical constraint properties in Concept HDL.



Important

Before modifying electrical constraint properties in the schematic, choose *Edit > Constraints > Update Schematic* in Concept HDL so that all the changes to constraints that you made in Constraint Manager are updated in the schematic. For more information on why this is required, see Exiting Constraint Manager on page 313.

The changes that you make in Concept HDL will be visible in Constraint Manager when you start Constraint Manager from Concept HDL.

Deleting Electrical Constraints

This section describes the procedures for deleting constraints in Constraint Manager and in Concept HDL.

Note: Cadence recommends that you delete constraints in Constraint Manager and not in Concept HDL.

See the following topics for more information on deleting constraints:

- Deleting Electrical Constraints in Constraint Manager on page 296
- Deleting Electrical Constraints in Concept HDL on page 296
- Buses and pin-pairs are treated as special objects when working with electrical constraints. For more information on working with electrical constraints on buses and pin-pairs, see:
  - Working with Electrical Constraints on Buses on page 275
  - Working with Electrical Constraints on Pin-Pairs on page 282
- See Working with Electrical Constraints in Hierarchical Designs on page 287 for recommendations on working with electrical constraints in hierarchical designs.

Deleting Electrical Constraints in Constraint Manager

If you have captured an electrical constraint in Constraint Manager or added an electrical constraint property in Concept HDL, you can delete the constraint in Constraint Manager.

To delete constraints in Constraint Manager, do the following:

1. Start Constraint Manager from Concept HDL.
2. Delete the constraint in Constraint Manager.
3. Choose *File > Save* in Constraint Manager or Concept HDL.
4. Choose *Tools > Constraints > Update Schematic* in Concept HDL.

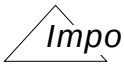
   If there is a constraint property or a placeholder for the constraint property in the schematic, the deletion will now be visible in the Hierarchy, Expanded, and Occurrence Edit modes in Concept HDL.

For more information on deleting constraints in Constraint Manager, see the documentation for *Constraint Manager*.

Deleting Electrical Constraints in Concept HDL

Cadence recommends that you delete constraints in Constraint Manager and not in Concept HDL.

You cannot delete constraint properties in Concept HDL when Constraint Manager is running. Exit Constraint Manager if you want to delete electrical constraint properties in Concept HDL.



If you delete an electrical constraint property on an object in the schematic, you must delete the constraint on the object in Constraint Manager also. If you do not delete the constraint in Constraint Manager, the constraint property will be backannotated to the schematic when you choose *Tools > Constraints > Update Schematic* or *Tools > Back Annotate* (with *Constraint Backannotation* check box selected) in Concept HDL. This will result in the electrical constraint property continuing to remain on the object in the schematic.

Auditing Obsolete Objects

A net or pin in Concept HDL that has an electrical constraint property or a placeholder for a constraint property becomes obsolete when you do any of the following:

- Delete the net
- Rename the net
- Rename a net to which the net is aliased or synonymed

Constraint Manager allows you to list the obsolete objects that no longer exist in Concept HDL but have electrical constraint information. You can delete the obsolete objects or merge the constraints on the obsolete object to an existing object. For example, if a net SELECT that has the PULSE_PARAM electrical constraint property becomes an obsolete object, you can merge the constraint property to an existing net in the schematic called CLOCK. The PULSE_PARAM electrical constraint property will visible on the net CLOCK in the schematic.

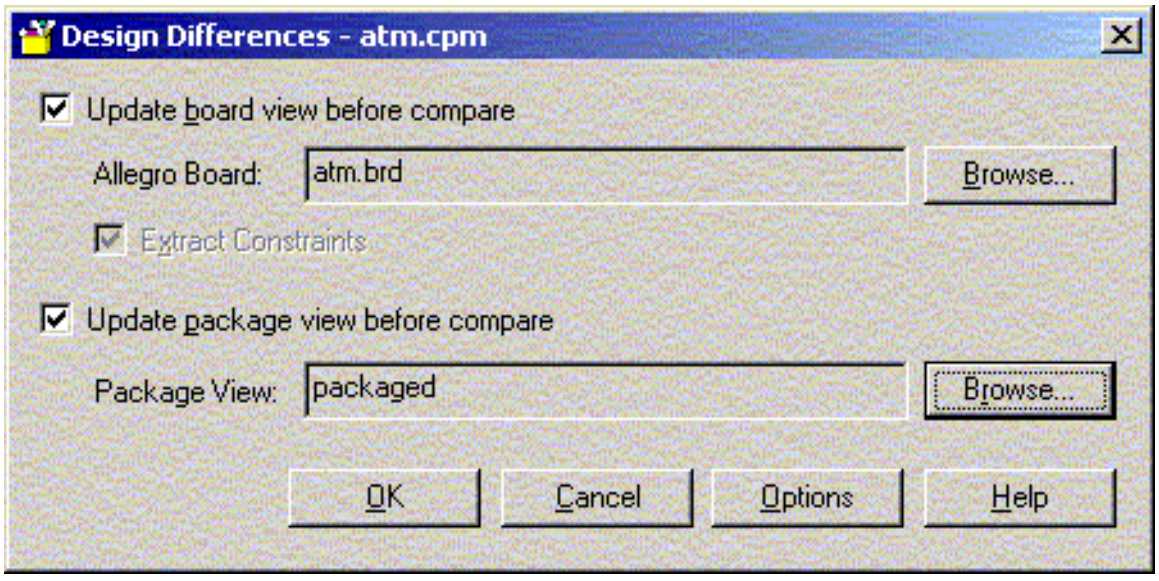
For more information on auditing obsolete objects, see the *Constraint Manager Design Guide*.

Viewing Constraint Differences between the Schematic and the Board

After running Export Physical, you can verify that the constraint information in the board matches the constraint information in the schematic by running the Design Differences tool:

1. Choose *Tools* > *Design Differences* in Concept HDL.

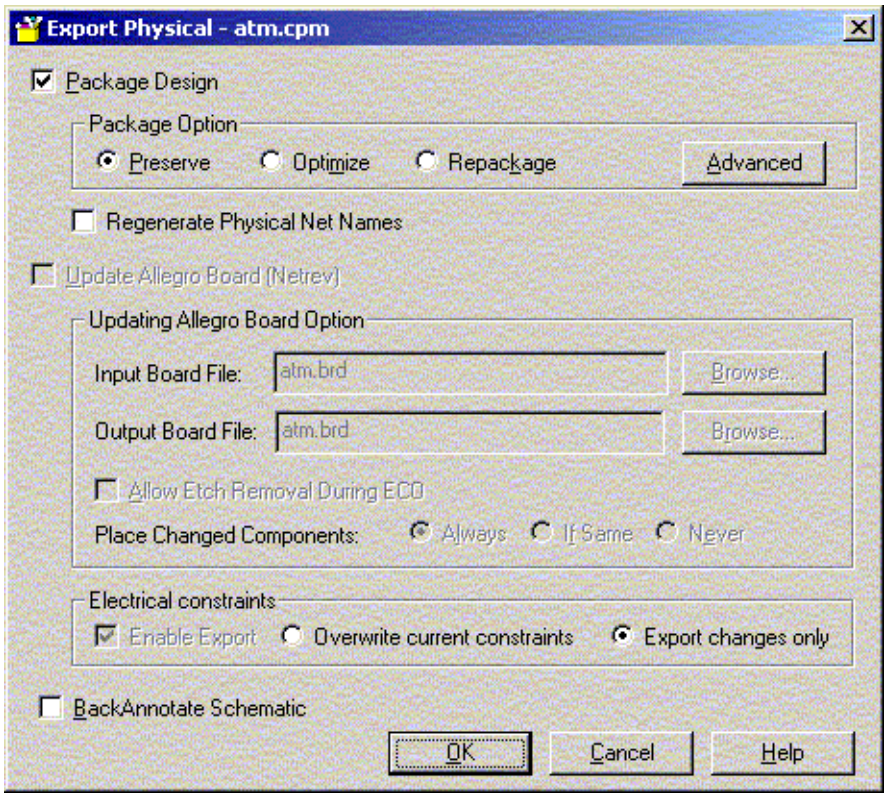
   The *Design Differences* dialog box appears.



The *Extract Constraints* check box is selected because you are using the Constraint Manager enabled flow. You cannot deselect this check box because once you are in the Constraint Manager enabled flow, you cannot go back to the traditional flow. For more information, see Constraint Manager Enabled Flow on page 263 and Traditional Flow on page 260.

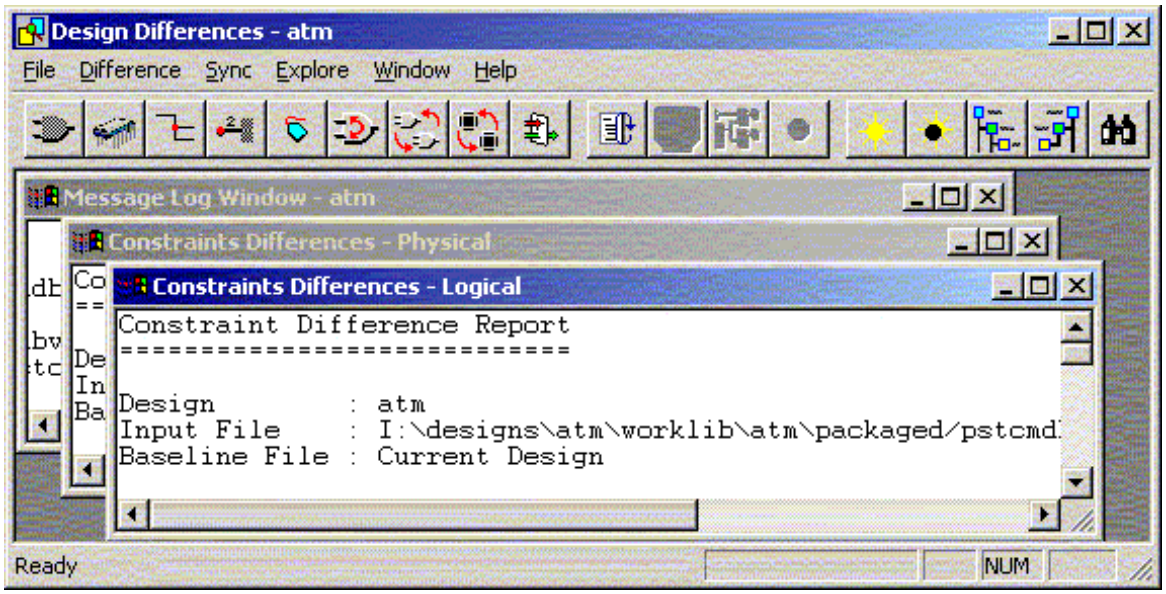
2. Click *OK*.

The *Export Physical* dialog box appears.



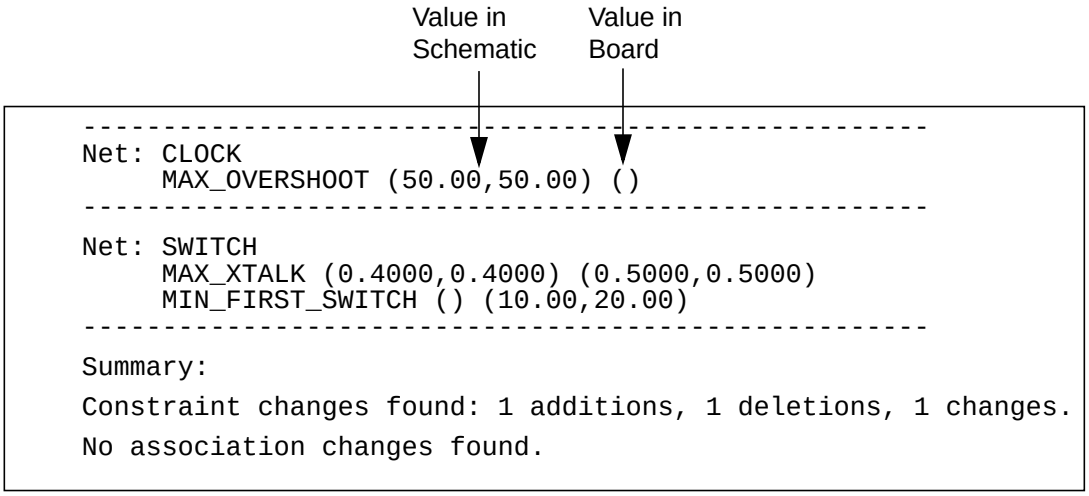
3. Click *OK*.

After Export Physical completes its operation, it passes the control back to the Design Differences tool. The *Design Differences* window appears displaying the *Constraints Differences-Logical* and *Constraints Differences-Physical* windows.



The Constraints Differences-Logical Window

The *Constraints Differences-Logical* window uses the schematic as the master to display the differences in electrical constraints information between the schematic and the board. For example, if the *Constraints Differences-Logical* window displays constraints differences as below:

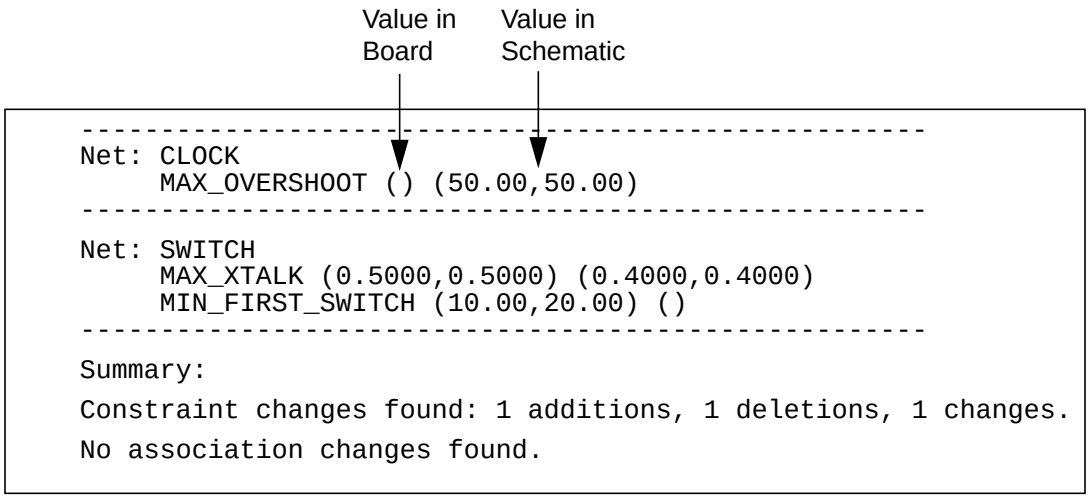


- The entry `MAX_OVERSHOOT (50.00,50.00) ()` for the net `CLOCK` indicates that the `MAX_OVERSHOOT` constraint exists on the net `CLOCK` in the schematic but does not exist on the net in the board. This means one of the following:
  - The constraint was added in the schematic after the last time the schematic and the board were synchronized.
  - The constraint was deleted in the board after the last time the board and the schematic were synchronized.
- The entry `MAX_XTALK (0.4000,0.4000) (0.5000,0.5000)` for the net `SWITCH` indicates that the value of the `MAX_XTALK` constraint on the net `SWITCH` in the schematic is `0.4000:0.4000` but the value of the constraint on the net in the board is `0.5000:0.5000`. This means one of the following:
  - The constraint was modified in the schematic after the last time the schematic and the board were synchronized.
  - The constraint was modified in the board after the last time the board and the schematic were synchronized.
- The entry `MIN_FIRST_SWITCH () (10.00,20.00)` for the net `SWITCH` indicates that the `MIN_FIRST_SWITCH` constraint does not exist on the net `SWITCH` in the schematic but exists on the net in the board. This means one of the following:
  - The constraint was deleted in the schematic after the last time the schematic and the board were synchronized.
  - The constraint was added in the board after the last time the board and the schematic were synchronized.
- The *Summary* section indicates that:
  - One constraint (`MAX_OVERSHOOT`) was added in the schematic.
  - One constraint (`MIN_FIRST_SWITCH`) was deleted in the schematic.
  - One constraint (`MAX_XTALK`) has changed.

The Constraints Differences-Physical Window

The *Constraints Differences-Physical* window uses the board as the master to display the differences in electrical constraints information between the board and the schematic. For

example, if the *Constraints Differences-Physical* window displays constraints differences as below:



- The entry `MAX_OVERSHOOT () (50.00,50.00)` for the net `CLOCK` indicates that the `MAX_OVERSHOOT` constraint does not exist on the net `CLOCK` in the board but exists on the net in the schematic. This means one of the following:
  - The constraint was deleted in the board after the last time the board and the schematic were synchronized.
  - The constraint was added in the schematic after the last time the schematic and the board were synchronized.
- The entry `MAX_XTALK (0.5000,0.5000) (0.4000,0.4000)` for the net `SWITCH` indicates that the value of the `MAX_XTALK` constraint on the net `SWITCH` in the board is `0.5000:0.5000` and the value of the constraint on the net in the schematic is `0.4000:0.4000`. This means one of the following:
  - The constraint was modified in the board after the last time the board and the schematic were synchronized.
  - The constraint was modified in the schematic after the last time the schematic and the board were synchronized.
- The entry `MIN_FIRST_SWITCH (10.00,20.00) ()` for the net `SWITCH` indicates that the `MIN_FIRST_SWITCH` constraint exists on the net `SWITCH` in the board but does not exist on the net in the schematic. This means one of the following:
  - The constraint was added in the board after the last time the board and the schematic were synchronized.

  - The constraint was deleted in the schematic after the last time the schematic and the board were synchronized.

- The *Summary* section indicates that:
  - One constraint (`MIN_FIRST_SWITCH`) was added in the board.
  - One constraint (`MAX_OVERSHOOT`) was deleted in the board.
  - One constraint (`MAX_XTALK`) has changed.

Synchronizing Electrical Constraints between Schematic and Board

If you are concurrently capturing constraints in both the schematic (using Concept HDL) and the board (using Allegro/APD or SPECCTRAQuest), you can resolve constraint modifications in both directions—schematic to board and board to schematic—using the Design Synchronization process.

Note: This section describes only the process for synchronizing constraints between the schematic and the board. For more information on the Design Synchronization process, see the *Design Synchronization and Packaging User Guide*.

The procedures for synchronizing constraints between the schematic and the board are described in the following sections:

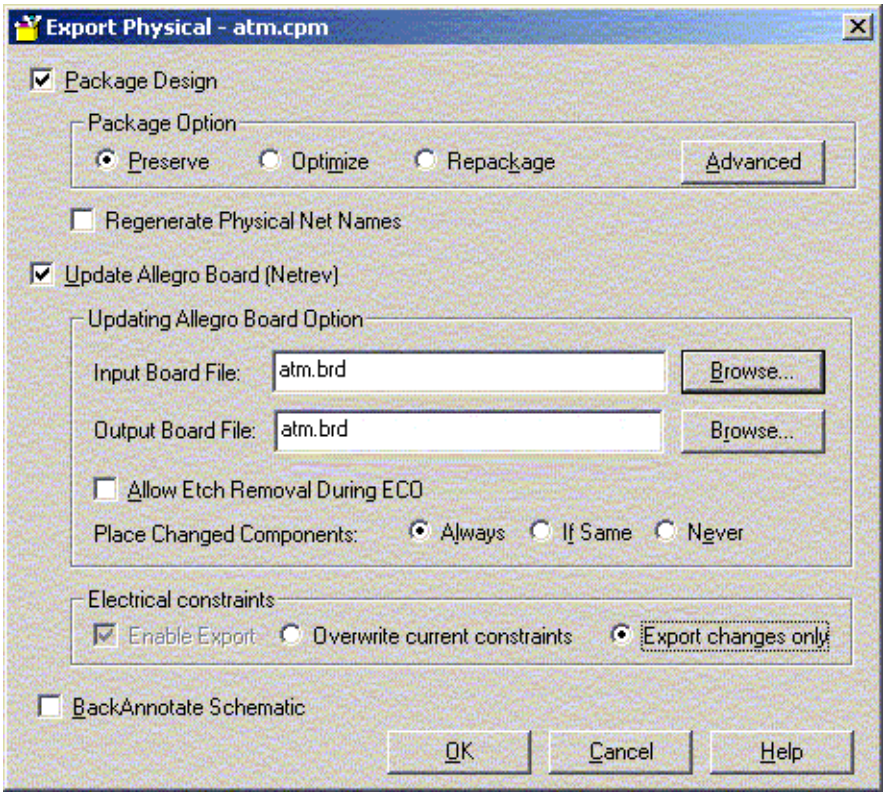
- Synchronizing Constraints in the Board with Constraints in the Schematic on page 302
- Synchronizing Constraints in the Schematic with Constraints in the Board on page 307
- Backannotating Constraints into the Schematic on page 311

Synchronizing Constraints in the Board with Constraints in the Schematic

When you start Constraint Manager from Allegro/APD or SPECCTRAQuest, the electrical constraints that you captured in Concept HDL will not appear in Constraint Manager. To synchronize the constraints in the board with the constraints in the schematic, you need to run Export Physical from Concept HDL.

1. Choose *File > Export Physical* from Concept HDL.

The *Export Physical* dialog box appears.



2. Select the *Package Design* check box. The design will be packaged when you run Export Physical.
3. Select the *Update Allegro Board (Netrev)* check box. All the changes (including changes in constraints) in the schematic will be exported to the board when you run Export Physical.

   Note: If you clear this check box and run Export Physical, you can update the schematic changes into the board later by choosing *File > Import > Logic* in Allegro or SPECCTRAQuest.

Caution

You must not select the Update Allegro Board (Netrev) check box when the layout designer is editing the board in Allegro/APD or SPECCTRAQuest.

4. Select the option for exporting constraints from the schematic to the board.

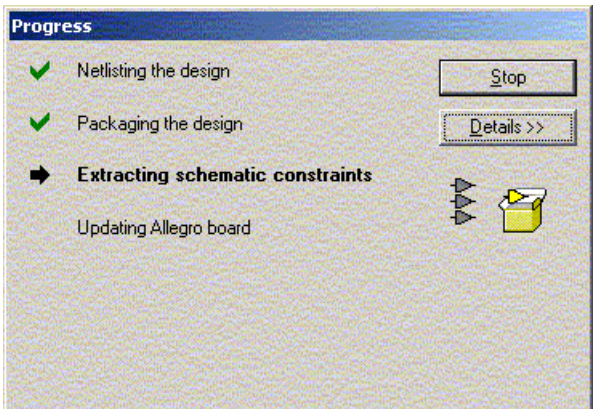
Select	To
Enable Export	This check box is selected by default if you are in the Constraint Manager enabled flow. For more information, see Constraint Manager Enabled Flow on page 263. This check box is not selected if you are in the traditional flow. For more information, see Traditional Flow on page 260.

Select	To
Overwrite current constraints	Delete all existing electrical constraint information in the Output Board File and replace it with the electrical constraint information currently available in the schematic. For example, suppose that you have: ■ `MAX_XTALK=0.5 mV` and `PULSE_PARAM=20 MHz` constraint on a net `INT` in the schematic ■ `MAX_XTALK=0.4 mV` and `MAX_OVERSHOOT=40 mV` constraints on the net `INT` in the board After you run Export Physical with the *Overwrite current constraints* option, the net `INT` in the board will have the the `MAX_XTALK=0.5 mV` and `PULSE_PARAM=20 MHz` constraints on it. Note that the `MAX_OVERSHOOT=40 mV` constraint on the net `INT` in the board has got deleted. This means the following: ■ If a constraint exists on a net in the schematic and the board, the constraint in the board is overwritten with the constraint in the schematic. ■ If a constraint exists on a net in the schematic but does not exist on the same net in the board, the constraint is added on the net in the board. ■ If a constraint does not exist on a net in the schematic but exists on the same net in the board, the constraint in the board gets deleted. *Important* Cadence recommends that you do not select this option if you are concurrently capturing constraints in the schematic and the board. This is because any changes that you made in the board will be deleted when you update the board.

Select	To
Export changes only	Export only the electrical constraint information that has changed in the schematic since the last export, and update such constraints in the Output Board File. For example, suppose that after the last time you ran Export Physical, you have: ■ `MAX_XTALK=0.4 mV` constraint on a net `INT` in the schematic ■ `MAX_XTALK=0.5 mV` on the net `INT` in the board Now, add the `MAX_OVERSHOOT=40 mV` constraint on the net `INT` in the schematic. After you run Export Physical with the *Export changes only* option, the net `INT` in the board will have the `MAX_XTALK=0.5 mV` and `MAX_OVERSHOOT=40 mV` constraints on it. Note that the value of the `MAX_XTALK` constraint on the net `INT` in the board has not changed. This means that only the constraint changes that you make in the schematic since the last time you ran Export Physical are updated in the board. Before running Export Physical, if you had changed the value of the `MAX_XTALK` constraint on the net `INT` in the schematic to `0.6 mV`, the net `INT` in the board will have the `MAX_XTALK=0.6 mV` constraint after you run Export Physical.

5. Select the *Backannotate Schematic* check box if you want to backannotate the latest packaged and constraints information to the schematic.
6. Click *OK*.

The *Progress* dialog box appears displaying the progress of the Export Physical process.



The design is netlisted and packaged. The files used for updating the constraint changes in the schematic to the board are created in the `packaged` view of the root design. The constraints in the schematic are extracted to a file called `pstcmdb.dat`. This file is used to update the constraints in the board.

The constraints in the board are synchronized with the constraints in the schematic.

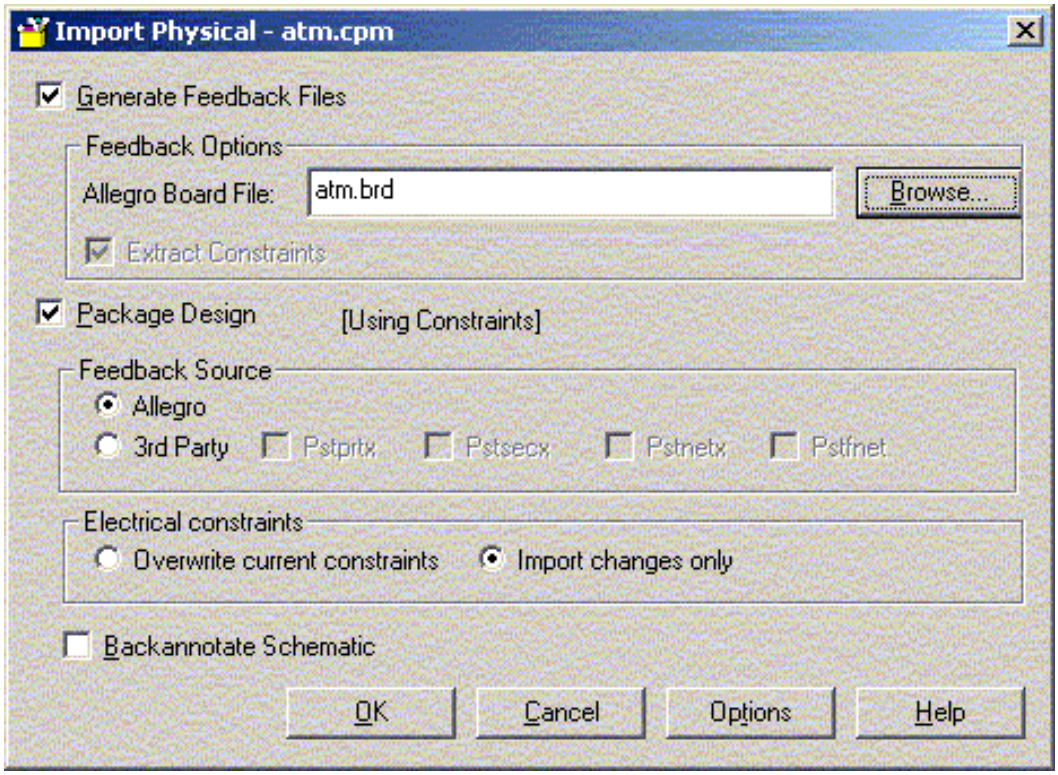
If you start Constraint Manager from Allegro/APD or SPECCTRAQuest, all the electrical constraints that you captured in Concept HDL will appear in Constraint Manager.

Synchronizing Constraints in the Schematic with Constraints in the Board

When you start Constraint Manager from Concept HDL, the electrical constraints that you captured in Allegro/APD or SPECCTRAQuest will not appear in Constraint Manager. For more information on capturing constraints in Allegro/APD or SPECCTRAQuest, see the documentation for *Allegro/APD*, *SPECCTRAQuest* and *Constraint Manager*. To synchronize the constraints in the schematic with the constraints in the board, you need to run Import Physical from Concept HDL and run backannotation.

1. Choose *File > Import Physical*.

The *Import Physical* dialog box appears.



The *Extract Constraints* check box is selected by default. This indicates that you are using the Constraint Manager enabled flow. For more information, see Constraint Manager Enabled Flow on page 263.

2. Select the *Generate Feedback Files* check box. When you run Import Physical, feedback files are created from the Allegro or SPECCTRAQuest board.
3. Select the *Package Design (Feedback)* check box and the *Allegro* option. When you run Import Physical, Packager-XL is run in the feedback mode using the feedback files from Allegro.

4. Select the option for exporting constraints from the board to the schematic.

Select	To
Overwrite current constraints	Delete all existing electrical constraint information in the schematic and replace it with the electrical constraint information currently available in the board. For example, suppose that you have: ■ `MAX_XTALK=0.5 mV` and `PULSE_PARAM=20 MHz` constraint on a net `INT` in the board ■ `MAX_XTALK=0.4 mV` and `MAX_OVERSHOOT=40 mV` constraints on the net `INT` in the schematic ■ After you run Import Physical with the *Overwrite current constraints* option, the net `INT` in the schematic will have the `MAX_XTALK=0.5 mV` and `PULSE_PARAM=20 MHz` constraints on it. Note that the `MAX_OVERSHOOT=40 mV` constraint on the net `INT` in the schematic has got deleted. This means the following: ■ If a constraint exists on a net in the board and the schematic, the constraint in the schematic is overwritten with the constraint in the board. ■ If a constraint exists on a net in the board but does not exist on the same net in the schematic, the constraint is added on the net in the schematic. ■ If a constraint does not exist on a net in the board but exists on the same net in the schematic, the constraint in the schematic gets deleted. *Important* Cadence recommends that you do not select this option if you are concurrently capturing constraints in the board and the schematic. This is because any changes that you made in the schematic will be deleted when you run backannotation in Concept HDL.

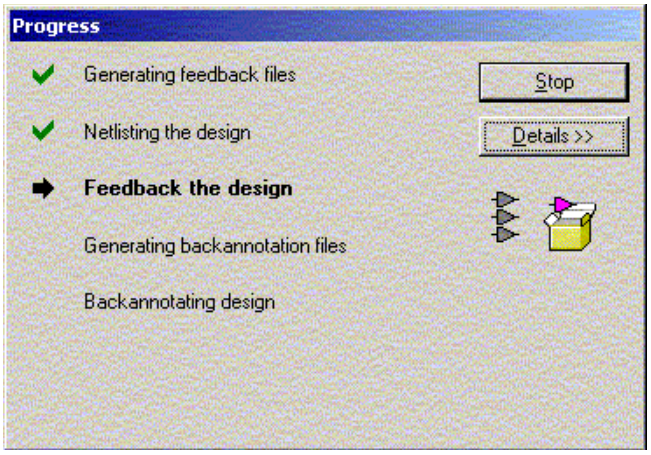
Select	To
Import changes only	Import only the electrical constraint information that has changed in the board since the last import, and update such constraints in the schematic. For example, suppose that after the last time you ran Import Physical, you have: ■ `MAX_XTALK=0.4 mV` constraint on a net `INT` in the board ■ `MAX_XTALK=0.5 mV` on the net `INT` in the schematic Now, add the `MAX_OVERSHOOT=40 mV` constraint on the net `INT` in the board. After you run Import Physical with the *Import changes only* option, the net `INT` in the schematic will have the `MAX_XTALK=0.5 mV` and `MAX_OVERSHOOT=40 mV` constraints on it. Note that the value of the `MAX_XTALK` constraint on the net `INT` in the schematic has not changed. This means that only the constraint changes that you make in the board since the last time you ran Import Physical are updated in the schematic. Before running Import Physical, if you had changed the value of the `MAX_XTALK` constraint on the net `INT` in the board to `0.6 mV`, the net `INT` in the schematic will have the `MAX_XTALK=0.6 mV` constraint after you run Import Physical.

5. Select the *Backannotate Schematic* check box if you want all the changes (including changes in constraints) in the board to be backannotated to the schematic when you run Import Physical.

   Note: Clear this check box if you do not want the schematic to be backannotated when you run Import Physical. You can perform backannotation later by choosing *Tools > Back Annotate* in Concept HDL. For more information, see Backannotating Constraints into the Schematic on page 311.

6. Click *OK*.

The *Progress* dialog box appears displaying the progress of the Import Physical process.



The feedback files are created from the Allegro or SPECCTRAQuest board. Packager-XL is run in the feedback mode using the feedback files from Allegro. The files used for backannotating the constraint changes in the board to the schematic are created in the `packaged` view of the root design. The constraints in the board are extracted to a file called `pstcmback.dat`. This file is used to backannotate the changes in constraints in the board to the schematic.

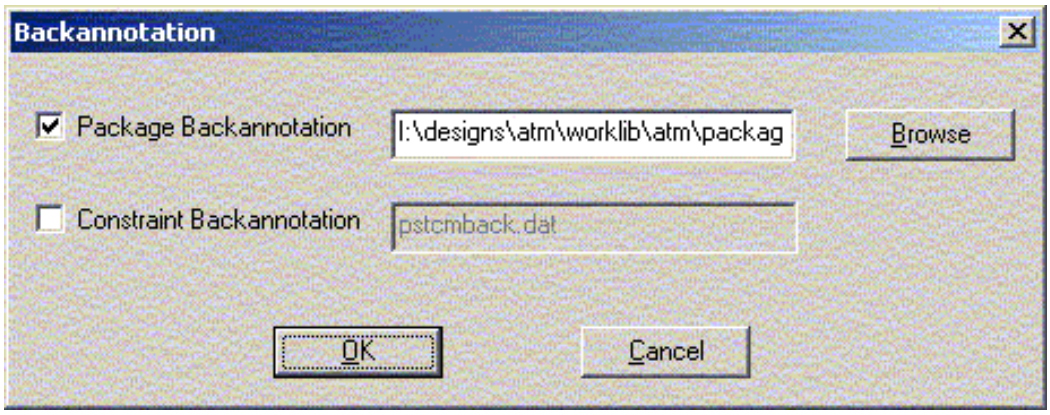
The constraints in the schematic are synchronized with the constraints in the board. When you start Constraint Manager from Concept HDL, all the electrical constraints that you captured in Allegro/APD or SPECCTRAQuest will appear in Constraint Manager.

Backannotating Constraints into the Schematic

If you do not select the *Backannotate Schematic* check box in the Import Physical dialog box, changes (including changes in constraints) in the board will not be backannotated to the schematic when you run Import Physical. You can run the backannotation process later by doing the following:

1. Choose *Tools* > *Back Annotate* in Concept HDL.

The *Backannotation* dialog box appears.



2. Select the *Package Backannotation* check box if you want to backannotate the changes (excluding changes in constraints) in the board to the schematic.
3. Select the *Constraint Backannotation* check box if you want to backannotate only the changes in constraints in the board to the schematic.
4. Click *OK*.

Cross Probing between Concept HDL and Constraint Manager

You can perform cross probing between Concept HDL and Constraint Manager to highlight nets.

To perform cross probing from Constraint Manager to Concept HDL

➤ In Constraint Manager, select a net and choose *Objects > Select*.

Switch to Concept HDL. The net is highlighted in Concept HDL.

To select multiple nets in Constraint Manager and highlight all of them together in Concept HDL, do the following:

1. In Constraint Manager, select multiple the nets by doing one of the following:
   - Press *Ctrl* and click to select individual nets in no particular order.
   - Press *Shift* and *click* to select a range of nets.
2. Choose *Objects > Select*.

Switch to Concept HDL. The nets are highlighted in Concept HDL.

To perform cross probing from Concept HDL to Constraint Manager

➤ In Concept HDL, select a net and choose *Display > Highlight*.

Switch to Constraint Manager. The net is highlighted in Constraint Manager.

To select multiple nets in Concept HDL and highlight all of them together in Constraint Manager, do the following:

1. Press *Ctrl* and click to select individual nets in no particular order.
2. Choose *Display > Highlight*.

Switch to Constraint Manager. The nets are highlighted in Constraint Manager.

Exiting Constraint Manager

➤ Choose *File > Exit* in Constraint Manager.

If you choose to save the changes that you made in Constraint Manager, the changes are updated in the schematic. These changes are visible in the schematic only in the Occurrence Edit mode. You must choose *Edit > Constraints > Update Schematic* in Concept HDL to view the changes in the Hierarchy or Expanded mode.

The changes that you made in Constraint Manager are saved in the Occurrence Property File. If you do not choose *Edit > Constraints > Update Schematic* after exiting Constraint Manager, and modify or delete an electrical constraint property in the schematic in the Hierarchy or Expanded mode in Concept HDL, the changes in the schematic will be lost. This is because when you start Constraint Manager from Concept HDL, it reads the constraints existing in the Occurrence Property File. For more information on the Occurrence Property File, see the *Packager-XL Reference*.

How Nets Are Displayed in Constraint Manager

If you add a bus CLOCK<3..0> in Concept HDL and start Constraint Manager before packaging the design, net names are displayed in Constraint Manager in the canonical (logical) format as below:

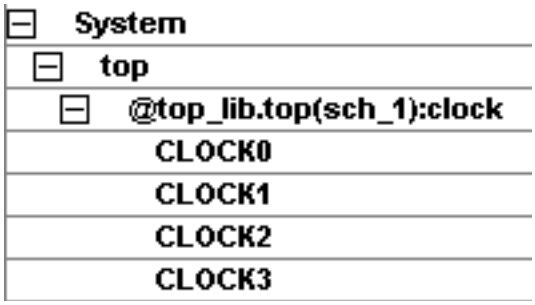
The entry @top_lib.top(sch_1):clock means that the bus CLOCK<3..0> is on the schematic view of the cell top. The cell top is located in the top_lib library in the project.

The four bits of the bus are displayed as shown below:

```
@top_lib.top(sch_1):clock(0)
@top_lib.top(sch_1):clock(1)
@top_lib.top(sch_1):clock(2)
@top_lib.top(sch_1):clock(3)
```

Note: Suppose that a net DATA is aliased to a net INT\BASE. Add a constraint property on the net DATA. Before you package the design Constraint Manager displays the constraint on the net DATA, even though the net INT is the base net. This is because DATA is lexicographically smaller than INT. After you package the design, Constraint Manager displays the constraint on the net INT because INT is the physical net name. For more information on declaring a net as the base net, see Declaring a Base Signal on page 138.

After you package and backannotate the design, the bus CLOCK<3..0> is displayed in Constraint Manager using physical (packaged) net names, as below:



To display the bus in Constraint Manager using logical net names, do the following:

1. Choose *View > Options* in Constraint Manager.

   The *View Options* dialog box appears.

2. Select the *Physical* option and click *Close*.

Files Created by Constraint Manager

The following files are created when you run Constraint Manager from Concept HDL.

- pstcmback.dat on page 315
- concept2cm.log on page 315

pstcmback.dat

This file contains information on the differences in electrical constraints existing in the board and the schematic. When you run backannotation, this file is used to backannotate the differences in constraints information to the schematic. This file is created in the `packaged` view of the root design.

concept2cm.log

This is the log file that contains messages, warnings, and errors when you start Constraint Manager from Concept HDL. This file is created in the `temp` directory under the project directory.

The `concept2cm.log` file contains the following information:

- Errors in the syntax of the value of electrical constraint properties on nets or pins in the schematic.
- Constraint Difference Report

Tip

Use Global Find (*Tools > Global Find*) in Concept HDL to quickly locate a net or pin that has an electrical constraint property with the wrong syntax.

12

Working with Block Designs

This section describes the procedures for using blocks to create hierarchical designs in Concept HDL.

About Blocks

Block diagrams let you create and edit symbols for top-level drawings. The symbols can then be replaced with functional designs.

All blocks have the property BLOCK=TRUE attached to the origin of the block.

These conventions apply to blocks:

- Show inputs on the left side of the block.
- Show outputs on the right.
- Any signal going through the top or bottom of a block defaults to an INOUT.

Note: Even if you have a BLOCK=TRUE property on a symbol, you cannot edit it. Concept HDL does not support this feature currently.

About View Generation in Hierarchical Designs

Genview lets you generate a design view from an existing view. A design can be represented by these views:

- Schematic (SCH)
- Symbol (SYM)
- VHDL
- Verilog

You can generate views

- Top down, where top-level symbol drawings are converted into VHDL or Verilog templates.
- Bottom up using a schematic or VHDL or Verilog text to create a symbol drawing.

Generating Views for Top-Down Design

After creating a top-level block diagram, you can create the corresponding VHDL or Verilog template. You can view or edit the templates by adding properties to the origin of a symbol that was created using *Tools > Generate View*. You can also add pin properties to change the default mode and type of ports in the VHDL or Verilog template.

Generating Views for Bottom-Up Design

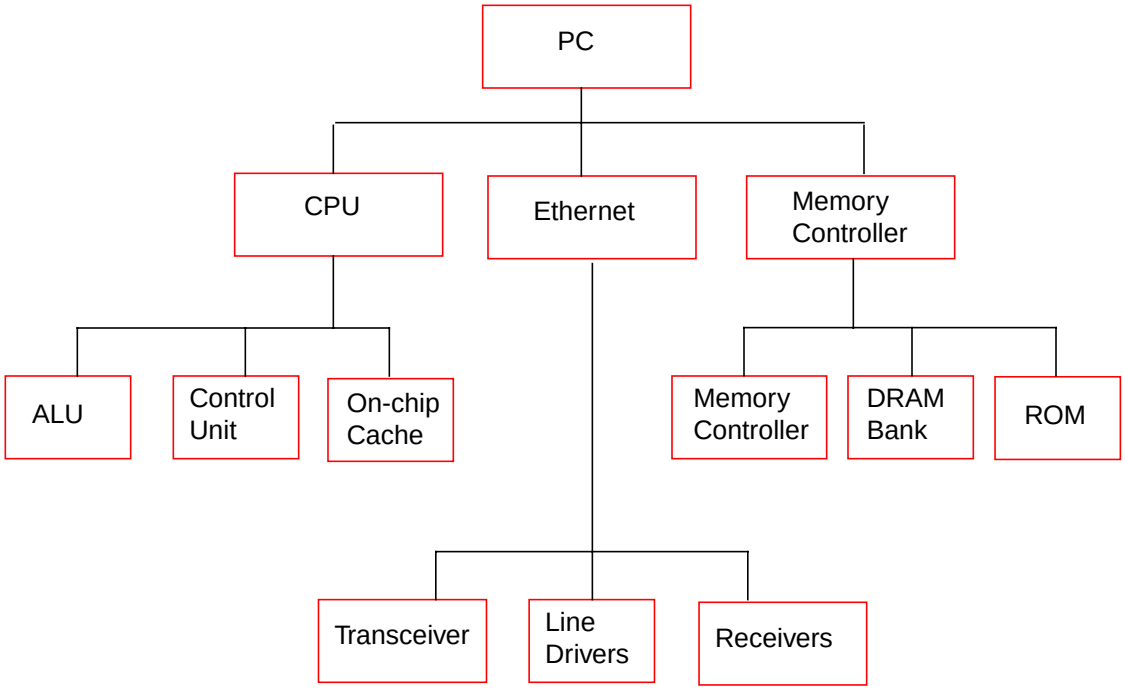
In a bottom-up design, you can create a symbol from a VHDL or Verilog template.

Values for VHDL and Verilog properties are obtained from the template file. Using the template lets you create a symbol with minimal editing and reduces errors from pin name mismatching.

Creating Hierarchical Designs

A hierarchical design is a large and complex design divided into sub designs. Each of the sub designs can be further divided into sub designs. For example, if you have a design called PC that contains sub-designs CPU, Ethernet, and Memory Controller, PC (the top-level design) is called a hierarchical design.

Example of a Hierarchical Design



The hierarchical design method is typically followed for large and complex designs. These designs are divided into individual modules where each module represents a logic function.

To create a hierarchical schematic in Concept HDL, you can choose either of the following methods:

- Top Down method
- Bottom Up method

Top Down Method

In the Top Down method, you first create the top-level drawing (PC in this case). In the top-level drawing, you can add blocks that represent individual modules. In the case of PC, the top-level drawing will has three blocks:

- CPU
- Ethernet
- Memory Controller

After creating the top-level drawing with the necessary blocks, you create the lower level schematics and save them as cells. These schematics should have the same names as those of the blocks in the top-level schematic.

For example, If the blocks in the schematic PC are named CPU, Memory, and Ethernet, the lower level schematics should be named CPU.SCH.1.1, Memory.SCH.1.1, and Ethernet.SCH.1.1 respectively. These names will ensure that Concept HDL links the schematics with the blocks. When you double-click on the block CPU, Concept HDL descends to CPU.SCH.1.1.

To create the hierarchical design

1. Create a top-level schematic. For this example, call it PC.SCH.1.1
2. Add three blocks to PC.SCH.1.1.

   Name the blocks CPU, MEMORY, and ETHERNET. Choose Block > Rename to rename the blocks.
3. Choose *File > New* to create a schematic for CPU.
4. Add the following blocks:
   - ALU
   - CU
   - OCC
5. Choose *File > Save As...* to access the *View Save As* dialog box.
6. In the *Library* field, choose *pc_lib*.
7. In the tree view, select CPU.
8. Select `Schematic` in the *View* field.
9. Click *Save*.

   The schematic you created for the CPU block is saved as CPU.SCH.1.1.

Similarly, you can create schematics using *File > New* and save them using *File > Save As* for ALU, CU, and OCC in CPU.SCH.1.1. After completing these tasks, use *File > Return* to return to the top-level drawing, PC.SCH.1.1. In PC.SCH.1.1, You can double-click on the CPU block to descend through the hierarchy.

You can complete the design PC by creating blocks and schematics for all levels in the design.

Bottom Up Method

In the Bottom Up method, you can create a low-level schematic first. For the design PC again, you can create the schematic drawings for ROM, DRAM Bank, and Memory Controller. Name the drawings ROM.SCH.1.1, DRAM.SCH.1.1, and MEM.SCH.1.1, respectively.

You then create the schematic for a higher-level drawing; for example, Memory Controller. Name the schematic Memory.SCH.1.1. In Memory.SCH.1.1, create three blocks and name them (*Block > Rename*) ROM, DRAM, and MEM. After saving the blocks, you can descend to MEM.SCH.1.1 by double clicking on block MEM in Memory.SCH.1.1.

Adding a Block

To add a block using the Concept HDL block name

1. Choose *Block > Add*.
2. Click where you want to place the block, move the cursor diagonally, and click again.

   Note: If you are zoomed too far in on the drawing, Concept HDL warns that blocks must have a minimum width and height. Zoom out to place the block.
3. Concept HDL assigns the name BLOCKn. You can change block names at any time by choosing *Block > Rename*.

To add a block and name it yourself

1. Choose *Block > Add*.
2. Click right and choose *Block Name…* from the pop-up menu.
3. Type a name in the *Block Name* box.
4. If you enter the name of an existing block, a copy of the specified block attaches to the cursor. Click where you want to place the block.

   If you enter a unique name, click where you want to place the block, move the cursor diagonally, and click again.

Note: If you are zoomed too far in on the drawing, Concept HDL warns that blocks must have a minimum width and height. Zoom out to place the block.

Renaming a Block

1. Choose *Block > Rename*.

2. Enter a block name in the *Block Name* box.
3. With the new block name attached to the cursor, click the block that you want to rename.

   Note: If you specify an existing block name, Concept HDL asks if you want to overwrite the existing block. Choose *Yes* or *No*.

Resizing a Block

1. Choose *Block > Stretch*.
2. Click a corner or side of the block that you want to stretch, move the cursor to resize the block, and click again.

Wiring Blocks

To manually draw a wire between blocks

1. Choose *Block > Draw Wire*.
2. Click the edge of one block.
3. Click wherever you want the wire to bend, or click the edge of another block.

Note: If no block pins exist where you want to add a wire, Concept HDL adds pins and names them PINn. You can change this name at any time by choosing *Block > Rename Pin*.

Tips for Wiring Blocks

- Click left to end a wire at a pin, dot, or other wire.
- Click left twice at the final point to end a wire in a free space.

Click Ctrl+left and continue clicking to change the bend of the wire.

To auto-route a wire between blocks

1. Choose *Block > Route*.
2. Click the edge of one block and then click the edge of another block.

   Note: If no block pins exist where you want to add a wire, Concept HDL adds pins and names them PINn. You can change pin names at any time by choosing *Block > Rename* Pin.

Note: You can also run the `route` command using this stroke pattern:

Mirroring Components or Blocks

1. Choose *Edit > Mirror*.
2. Click a component or block.

Displaying Block Properties

All blocks have a BLOCK=TRUE property attached to the block's origin. By default, this property is not displayed. This property distinguishes a block component from a body component.

To display the BLOCK=TRUE property

1. Display the console window, and enter the command display both.
2. Select a block.

The BLOCK=TRUE property appears near the origin of the block.

Adding Block Pins

1. Choose *Block > Add Pin*.
2. Choose the type of the pin you want to add.

   The *Block Pin Add* dialog box appears.
3. Type one or more pin names on separate lines.
4. Click the edge of the block in the same order that you entered pin names.

   Concept HDL adds the pin(s) where you specify.

Note: To toggle the pin type before you place the pin on the block, click the right mouse button and choose *Change Mode*. Alternately, press *Ctrl* and click the left mouse button in a two-button mouse or click the middle mouse button in a three-button mouse.

Renaming Block Pins

1. Choose *Block > Rename Pin*.

   The *Block Pin Rename* dialog box appears.
2. Type one or more pin names on separate lines.
3. Select existing pins that you want to rename.

Concept HDL changes the name of the pin(s) you select.

Deleting Block Pins

1. Choose *Block > Delete Pin*.
2. Click the pins you want to delete.

Note: Click the pin, not the pin name.

Moving Block Pins

1. Choose *Block > Move Pin*.
2. Click the block pin that you want to move.
3. Click the pin's new location.

Note: You cannot move a pin across components.

Using Read-only Blocks in Your Design

You can create a reusable block using Concept HDL and Allegro and maintain it in a reference library. You can then use the block as a read-only block in other designs. For more information on creating reusable blocks and using them in other designs, see the "Design Reuse" chapter of the *Packager-XL Reference*.

Navigating the Drawing Hierarchy

To view a block diagram from the top-level schematic

1. Choose *File > Edit Hierarchy > Descend*.
2. Click a block in the schematic.

   Concept HDL descends into the symbol view.
3. Continue descending the drawing hierarchy by repeating steps 1 and 2.

To descend into drawings while in In Hierarchy mode, you can set the environment variable CONCEPT_DESCEND_EDIT_LIST variable.

Example

If you have vlog_rtl, sch_1, and sym_1 views of the drawing and wish to descend into them when you double-click on the top level drawing, set the following environment variable.

```
Setenv CONCEPT_DESCEND_EDIT_LIST vlog_rtl, sch_1, sym_1
```

After setting this environment variable, when you double-click on the drawing, Concept HDL searches for the vlog_rtl view and displays it. If this view is not present, Concept HDL displays the sch_1 view.

To ascend the drawing hierarchy from a lower level block diagram

1. Choose *File > Edit Hierarchy > Ascend*.
2. Continue ascending the drawing hierarchy by repeating steps 1 and 2.

To return to the previous drawing

➤ Choose *File > Return*.

Note: You can view a list of the drawings that Concept HDL will return to and the order in which they will be accessed by choosing *Display > Return*.

Generating a Design View

1. Choose *Tools > Generate View*.

   The *Genview* dialog box appears.

2. Specify the source view in the `lib.cell:view` format.

   You can also specify a Verilog or VHDL source file from which you want to generate the view.

   - ❑ Select *Verilog* in the *Type* drop-down list if you have selected a Verilog source file.
   - ❑ Select *VHDL* in the *Type* drop-down list if you have selected a VHDL source file.

3. If the source is a file, select the destination library where you want Concept HDL to create the destination cell.

   If the source is a view, the destination library is the same as the library for the source view.

4. Select the view that you want to generate in the *View* drop-down list.

5. In the *Type* drop-down list, select the type of the view you have selected in the *View* drop-down list:

Select the type	If you have selected the following view
Schematic	sch_1
Symbol	sym_1
VHDL	vhdl_1
Verilog	vlog_1

6. Select the *Retain Graphics* check box if you want to retain the placement of pins that already existed on the graphic for the symbol.

   For example, suppose that the symbol view already exists. If you add or delete a pin in the source view or source file and regenerate the symbol view, the placement of the pins that already existed (pins that were not deleted in the source view or source file) on the symbol will be retained.

   Cadence recommends that you use this option if you have already used the symbol on your schematic. This will ensure that the connectivity between a wire and a pin of the symbol on the schematic is not lost because the placement of the pin on the symbol does not change.

   If you do not select this check box, the graphic for the symbol is regenerated and the pin placement is done by Concept HDL using its internal algorithms.

7. Select the *Split Vector Ports* check box if you want the vectored ports in the source view or source file to be split into multiple pins (representing each bit of the vectored port) on the symbol.

   For example, if the source view or source file has a vectored port `DATA<3..0>`, the following four pins will be added on the symbol:

   - DATA<3>
   - DATA<2>
   - DATA<1>
   - DATA<0>

   If this check box is not selected, the symbol will have a pin named `DATA<3..0>`.

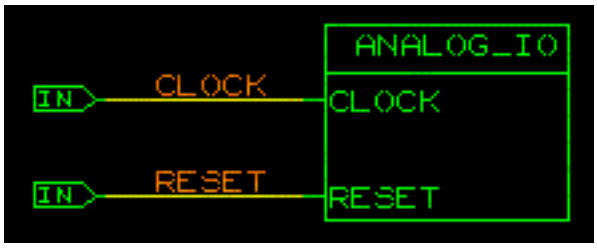
8. Click *Generate*.

   The *Output* field displays the results of the generate view process.

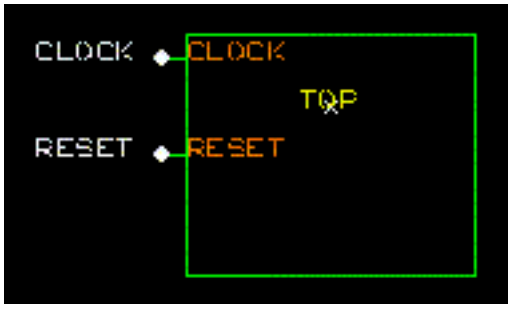
Example of Using Retain Graphics and Split Vectored Ports Options

The *Retain Graphics* and *Split Vectored Ports* options are explained below using an example.

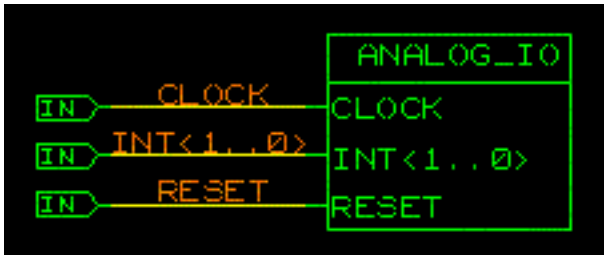
Suppose that you have a schematic `TOP.SCH.1.1` as below:



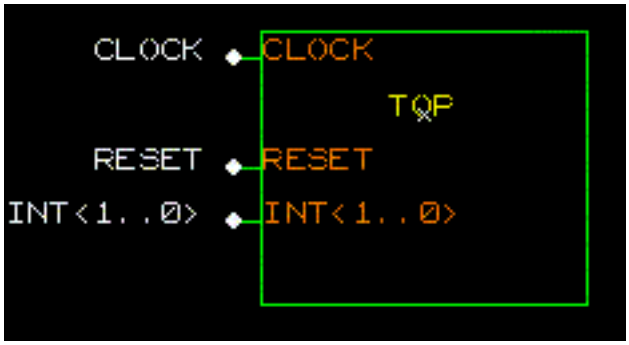
1. Generate the symbol for the schematic. A symbol named `TOP` will be created as below:



2. Add a pin `INT<1..0>` on the `ANALOG_IO` block and connect it to an input port `INT<1..0>` on the schematic, as below:



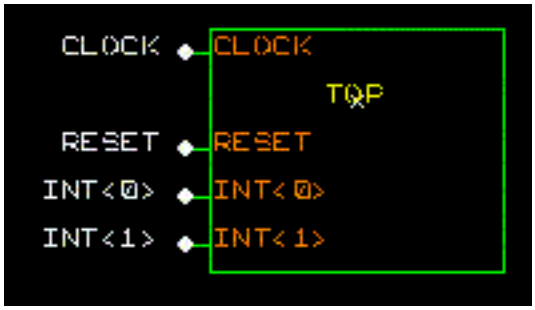
3. Generate the symbol for the schematic again with the *Retain Graphics* check box selected. The symbol TOP will be created as below:



Note that the placement of the pins CLOCK and RESET on the symbol have not changed.

Suppose that you have instantiated the symbol TOP on some other schematic page and have connected a wire to the pin CLOCK on the symbol. The connectivity between the wire and the pin CLOCK is not lost now because the placement of the pin on the symbol has not changed.

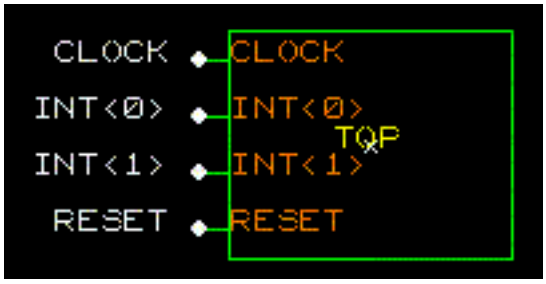
4. Generate the symbol for the schematic again with the *Retain Graphics* and *Split Vector Ports* check boxes selected. The symbol TOP will be created as below:



Note that the vectored port INT<1..0> in the schematic has been split into two pins (representing each bit of the vectored port) INT<0> and INT<1> on the symbol. The vectored pin INT<1..0> is deleted from the symbol, and the pins INT<0> and INT<1> are added as new pins on the symbol.

Also note that the placement of the pins CLOCK and RESET on the symbol have not changed.

5. Generate the symbol for the schematic again with the *Retain Graphics* check box deselected and the *Split Vector Ports* check box selected. The symbol TOP will be created as below:



Note that the placement of the pins on the symbol have changed. This is because the graphic for the symbol is regenerated when the *Retain Graphics* check box is not selected. The placement of the pins on the symbol is done by Concept HDL using its internal algorithms.

Suppose that you have instantiated the symbol TOP on some other schematic page and connected a wire to the pin CLOCK on the symbol. The connectivity between the wire and the pin CLOCK is lost now because the placement of the pin on the symbol has changed.

Adding a Symbol with Physical Part Information

1. Display the Component Browser.
2. Select a component in the Cells section of the Component Browser.

   If physical information is available for that component, physical part names are listed in a pop-up menu. If no PPT is found for a part, you can place a symbol that serves as a place holder for physical information which you can add later.
3. Select a part name.

   The Physical Part Filter for that part appears.
4. Select a part in the Physical Part Filter.
5. Click in the drawing to place the part.

13

Working with Groups

A group is a collection of schematic objects. These objects can be symbols, properties, notes, wire segments, and dots. Groups can be used to group objects on the same schematic page. Pins cannot be added to a group.

For each defined group, Concept HDL assigns a letter of the alphabet as the name of the group. The current group name is displayed in brackets in the *Group* menu. Group contents are listed in the *Group Contents* dialog box that appears when you choose *Group > Show Contents*.

Concept HDL names each group you define with consecutive letters: a, b, c, and so on. You can have up to 26 groups at one time in each drawing. If you define more than 26 groups, the group name resets back to *A*, and the newly defined group *A* replaces the previously defined one.

Note: When defining a group using the *Group By Rectangle* or the *Group By Polygon* menu options to include symbols, do not specify the points of the group on the symbol.

Creating a Group by Rectangle

When you create a group, you can either use Concept HDL's group identifier or assign the group identifier yourself. If you assign the group identifier, you must set the current group before proceeding with these steps.

1. Choose *Group > Create > By Rectangle*.
2. Click in the upper left corner, and then in the lower right corner of the rectangle you want to use to define the group.
3. Move the cursor diagonally from where you clicked first.

   A rectangle encloses the objects to be included in the group.
4. Click again.
5. Objects in the group are highlighted.

Creating a Group by Polygon

When you create a group, you can either use Concept HDL's group identifier or assign a group identifier. If you assign the group identifier, you must set the current group before proceeding with these steps.

1. Choose *Group > Create > By Polygon*.
2. Click near an object you want to group.

   The point you click on will be the starting point of the polygon you will draw to define the group. The objects you want to group should lie inside the borders of the polygon.
3. Continue clicking to define a polygon that encloses objects to be included in the group.
4. Click right and choose *Close Polygon* from the pop-up menu to close the polygon.

   Objects in the group are highlighted.

Creating a Group by Specifying an Expression

When you create a group, you can either use Concept HDL's group identifier or assign the group identifier yourself. If you assign the group identifier, you must set the current group before proceeding with these steps.

1. Choose *Group > Create > By Expression*.

   Concept HDL places all occurrences of objects matching the pattern into a group.
2. Enter a pattern string.

   The pattern is used to match component names, notes, property names, property values, or signal names. Properties can also be searched for by specifying both the name and the value separated by an equal sign.

   Wildcards are allowed in the pattern. An asterisk matches any number of characters and a question mark matches any single character. The pattern is not case sensitive.

   For example, let us suppose that a schematic includes the components LS00, LS03, LS04, and LS06. To include these components in a group, choose *Group > Create > By Expression*. In the *Pattern* dialog box, enter ls0*. This will include the above components in a group.
3. Choose *Group > Create > Next* to traverse to each object found in the search.

Creating a Group by Including Objects

1. Choose *Group > Create > Include*.
2. Select one object after another to include them in the group.

Grouping the Entire Schematic

1. Choose either *Group > Create > By Rectangle* or *Group > Create > By Polygon*.
2. Click right and choose *All* from the pop-up menu.

Including Additional Objects in a Group

1. Choose *Group > Show Contents*.
2. In the *Group Contents* dialog box, select a group *A* through *Z*.
3. Click *Show* under *Highlight*.

   The contents of the specified group are highlighted.
4. Choose *Group > Create > Include*.
5. Select an object to include it in the active group.

   The active group is shown in the *Group* menu options and on the *Group* toolbar.
6. In the *Group Contents* dialog box, reselect the group to confirm that the object was included.

   The object you included should appear highlighted.

Excluding Objects from a Group

1. Choose *Group > Show Contents*.
2. In the *Group Contents* dialog box, select a group *A* through *Z*.
3. Click *Show* under *Highlight*.

   The contents of the specified group are highlighted.
4. Choose *Group > Create > Exclude*.
5. Select an object to exclude from the specified group.

6. In the *Group Contents* dialog box, reselect the group *A* through *Z* to confirm that the object was excluded.

   The object you excluded should no longer appear highlighted.

Setting the Current Group

1. Choose *Group > Set Current Group*.
2. Specify a group name.
3. Click *OK*.

The group name you specify appears in the square brackets next to several group menu commands. The group menu is also displayed on the *Group* toolbar.

Viewing Group Contents

1. Choose *Group > Show Contents*.
2. In the *Group Contents* dialog box, select a group *A* through *Z*.
3. Click *Show* under *Highlight*.

   The contents of the specified group are highlighted. Click *Clear* to turn off highlighting for the group.

Moving a Group

1. Set the current group.
2. Choose *Group > Move [x]*.
3. Select the group to move.

   By default, attachments to entries in the group do not move. Choose *Change Attachment*.
4. Click right and choose *Change Attachment* from the pop-up menu to change attachments to either move with grouped objects or not at all.
5. Click at the new location.

Rotating a Group

1. Set the current group.
2. Choose *Edit > Rotate*.
3. Press the middle mouse button on a three-button mouse or *Ctrl*+left mouse button on a two-button mouse and click on one of objects in the group to rotate the group.

Rotating a Group of Properties

You can rotate a group that consists only of properties. The properties are rotated in place and not as a whole group.

1. Set the current group.
2. Choose *Edit > Rotate*.
3. Press the middle mouse button on a three-button mouse or *Ctrl*+left mouse button on a two-button mouse and click on one of properties in the group to rotate the properties in the group.

Spinning a Group

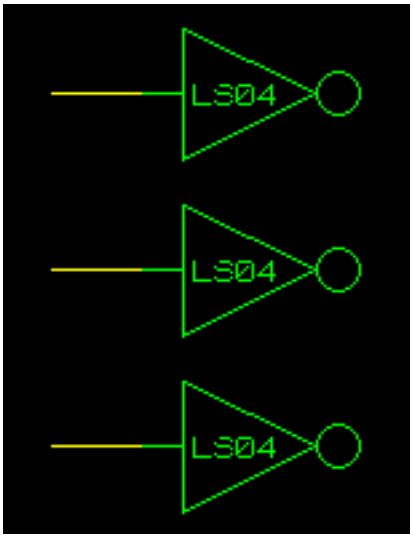
1. Set the current group.
2. Choose *Edit > Spin*.
3. Press the middle mouse button on a three-button mouse or *Ctrl*+left mouse button on a two-button mouse and click on one of objects in the group to spin the group.

Mirroring a Group

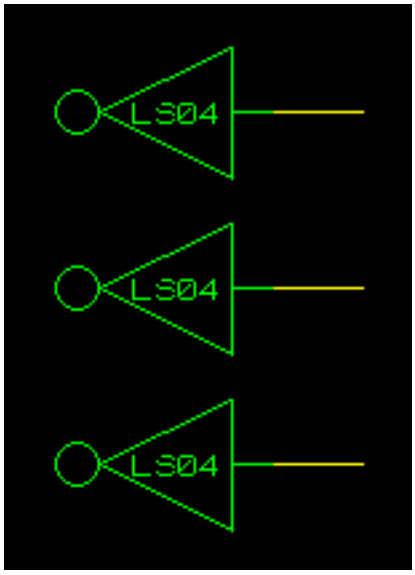
1. Set the current group.
2. Choose *Edit > Mirror*.
3. Press the middle mouse button on a three-button mouse or *Ctrl*+left mouse button on a two-button mouse and click on one of objects in the group.

   The group is attached to the cursor for you to place on the drawing.
4. Click in the same drawing or in another window to place the mirrored group.

If the components in the group are in a horizontal position, the group is mirrored horizontally. For example, the three `ls04` components in a group are in a horizontal position.



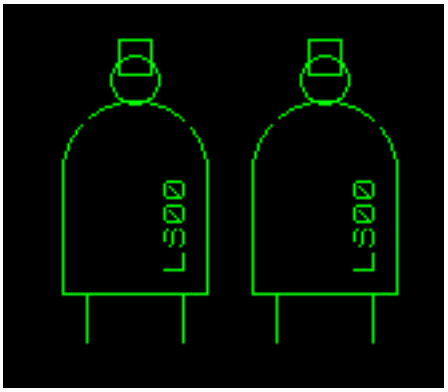
If you mirror this group, it will be mirrored horizontally, as below:



If the components in the group are in a vertical position, you can mirror the group vertically by doing the following:

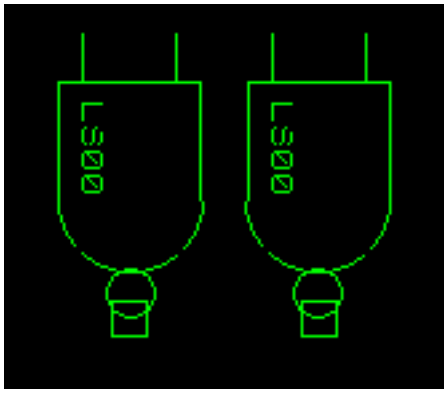
1. Set the current group.

For example, the two `ls00` components in a group are in a vertical position.

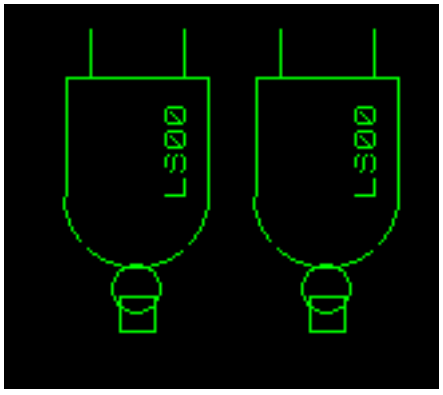


2. Rotate the group twice (by 180 degrees).

   For more information, see Rotating a Group on page 335.



3. Choose *Edit > Mirror*.
4. Press the middle mouse button on a three-button mouse or *Ctrl*+left mouse button on a two-button mouse and click on one of objects in the group.



Copying a Group

1. Set the current group.
2. Choose *Group > Copy [x]*.

   The group is attached to the cursor for you to place on the drawing. To copy an object with its properties, click right and choose *All* from the pop-up menu.
3. Click in the same drawing or in another window to place the copy.
4. If you're done placing copies but wish to remain in *Group > Copy* mode, choose *Terminate Selection* from the pop-up menu. To exit *Group > Copy* entirely, choose Done from the pop-up menu.

To copy a group and its properties:

1. Set the current group.
2. Choose *Group > Copy All [x]*.

   The group is attached to the cursor for you to place on the drawing.
3. Click in the same drawing or in another window to place the copy.
4. When you have finished placing copies but wish to remain in *Group > Copy All* mode, choose *Terminate Selection* from the pop-up menu. To exit *Group > Copy All* entirely, choose *Done* from the pop-up menu.

Note: You can also copy properties in a group when you choose *Group > Copy* or *Group > Array*, click right, and choose *All* from the pop-up menu.

To make multiple copies of a group:

1. Set the current group.
2. Choose *Group > Array [x]*.

   The group is attached to the cursor for you to place on the drawing. To copy an object with its properties, click right and choose *All* from the pop-up menu.
3. In the *Array Size* dialog box, specify the number of copies to make, and click *OK*.
4. Click in the same drawing or in another window to place the copies.
5. If you're done placing copies but wish to remain in *Group > Array* mode, choose *Terminate Selection* from the pop-up menu. To exit *Group > Array* entirely, choose *Done* from the pop-up menu.

Deleting a Group

1. Set the current group.
2. Choose *Group > Delete [x]*.

   Concept HDL deletes all objects in the specified group.

Specifying Color for a Group

1. Set the current group.
2. Choose *Group > Color [x]*.

   The *Color* toolbar is displayed.
3. Select a color from the toolbar.

   Concept HDL colors all objects in the group according to your selection.

Highlighting a Group on the Schematic

1. Set the current group.
2. Choose *Group > Highlight [x]*.

   Concept HDL highlights all objects in the specified group.

Replacing Components in a Group

1. Set the current group.
2. Choose *Group > Components > Replace*.

   The *Replace Component* dialog box appears.
3. Select the component that should replace all components in the current group.

   If you want to replace the components in the group with a component along with its physical properties, do the following:

   a. Click *Physical*.

      The *Physical Part Filter* dialog box appears.

b. Select the appropriate row of physical properties from the *Physical Part Filter* dialog box and click *Close*.

All the components in the current group are replaced with version 1 of the component that you selected in the *Replace Component* dialog box.

Replacing Component Symbols in a Group (Versioning)

1. Set the current group.
2. Choose *Group > Components > Version*.

   The version number of all components in the group will be incremented by 1. If the new version does not exist, the original will be replaced. If a component has two versions and the second is being used, this will change it to the first version.

Breaking Up Components in a Group

1. Set the current group.
2. Choose *Group > Components > Smash*.

   Concept HDL breaks all components in the group into individual elements such as lines, arcs, and dots.

Specifying Property Display for a Group

1. Set the current group.
2. Choose *Group > Property Display > Name/Value/Both/Invisible*, depending on whether you want to

- display only the property Name
- display only the property Value
- display Both the property name and value
- make properties Invisible

Changing the Text Size in a Group

To change the size of the text (property name, property value, signal name, note) of the objects in a group, use the following console command:

```
textsize <size in inches> <group name>
```

Note: You can specify a text size that has up to three decimal places. The minimum text size that you can specify is 0.008 inches and the maximum is 1.740 inches. The text size you specify should be a multiple of 0.002 inches.

For example, to change the size of all the text in a schematic page to 0.96 inches, create a group, say A, that covers all the objects in the schematic page (see Grouping the Entire Schematic on page 333) and enter the following command in the console window:

```
textsize 0.96 A
```

For more information on the `textsize` console command, see Textsize on page 600.

Modifying Components with the Same Part Name in a Group

If a group contains components with the same part name, you can modify the properties of all the components using *Group* > *Components* > *Modify*.

To modify the components with the same part name,

1. Set the current group.
2. Choose *Group* > *Components* > *Modify*.

   The *Physical Part Filter* appears.
3. Select the new row of properties and click *Close*.

   Concept HDL replaces the existing components with the new selection along with the key properties.

14

Working with Designs

This section describes the following tasks:

Expanding Your Design on page 343

Finding Nets and Cells in Your Design on page 344

Navigating Nets in Your Design on page 346

Running Scripts on page 347

Highlighting (Cross-Probing) Objects on page 350

Expanding Your Design on page 343

Applying Connectivity Changes from the Physical Design to Your Schematic

Back Annotating Your Design

Module Ordering on page 351

Displaying and Working with Schematic Page Numbers on page 358

Expanding Your Design

When you start a design, it is in hierarchy. This means that multiple drawings in a design are connected in a tree-like fashion. Before expansion, schematic views are obtained from the default specified in Project Manager Setup (in the Schematic box of the *Views* tab). Once a design is expanded, you can navigate up and down the drawing hierarchy using the Hierarchy Editor or by clicking on the Descend or Ascend icons as needed.

Expanding the design enables communications between Concept HDL and other tools and lets you associate an expansion configuration with the design.

When you create an expansion configuration using the Hierarchy Editor (*Tools > Hierarchy Editor*) and then expand the drawing (*Tools > Expand Design*), schematic views are obtained from the current configuration and not from the default specified in the Project Manager Setup.

With design expansion, Concept HDL reads all levels of the design so that you can:

- Highlight the objects (cross-probe) between tools using *Display > Highlight* and *Display > Dehighlight*.
- View properties from net synonyms that are in the current drawing or in other drawings.
- Navigate the drawing hierarchy based on the current expansion configuration.
- Find all synonyms of a specified net or all instances of a specified cell in a hierarchical design or multipage schematic.

To expand a design

➤ Choose *Tools > Expand Design*.

Concept HDL expands the design to read all pages and levels and to enable communication with other tools.

Note: Drawing expansion takes some time to complete, depending on the complexity of your design.

Finding Nets and Cells in Your Design

To find nets and cells:

1. Choose *Tools > Expand*.
2. Choose *Tools > Global Find*.

   The *Global Find* dialog box appears.
3. In the *Name* box, do one of the following:
   - ❑ Type the name of the net or the cell to be located. For example, typing ls04 locates all instances of the part ls04.

     Note: To find a vectored signal `DATA[3..0]` or `DATA(3..0)`, type `DATA` or `DATA<3..0>` in the *Name* box.
   - ❑ Select a previously entered name from the list.

4. Select the object type to be located as either *Net* or *Cell*.
5. To optionally restrict the search by property, do both of the following in the *With Property* section of the dialog box:
   - ❑ Type or select a property name in the *Name* box.
   - ❑ Type a property value in the *Value* box or type a * (asterisk) to locate all objects having the specified property name and any property value.

   When you enter the property name and the value, they are added to their respective list boxes so that they can be reused during the same design session.
6. Select how you want the search results to be listed - by full Hierarchical Names or by Library Location.
7. Click *Find* to begin the search.

   The search process begins. The *Find* button becomes the *Stop Find* button, which you can use to cancel a lengthy search in progress. You can also click *Close*.

   A message at the bottom of the dialog box tells you how many instances were found. The (unlabeled) status area box displays the instances of the object found.
8. Specify how you want a selected search result to be viewed: *Zoom to Object, Navigate* or both.
9. Click on a search result to view it in your design.

   When you select a result to view, the page containing the object appears with the object highlighted. If you chose Navigate, the *Global Navigate* dialog box appears so that you can move across the design to view all net instances listed in the search results box.
10. Click on another search result to view in the design.

    or

    Search for a different net or cell by entering a new net or cell name in the *Name* box.

    or

    Click *Close*.

Objects Not Found by Global Find

- Objects with the following properties:

VHDL_SLICE	VHDL_CONCAT	TIE

HDL_SCHEMATIC	HDL_POWER	HDL_CONCAT
HDL_REPLICATE	COMMENT_BODY	PLUMBING_BODY
FLAG_BODY	PINNAME_BODY	HDL_PORT
VHDL_PORT	VHDL_ALIAS	

- Objects named:

PIN NAMES	DRAWING	DEFINE
MENU		

- Path properties
- Signals with names having step size

  For example, A<31..0:2> is not found by Global Find.

Navigating Nets in Your Design

1. Choose *Tools* > *Expand Design*.
2. Choose *Tools* > *Global Navigate*.

   The *Global Navigate* dialog box appears.
3. Select how the search results are to be viewed - by full Hierarchical Names or by Library Locations.
4. Select a net in your design.

   The hierarchical name for the net appears in the Hierarchical Names box, and the message box indicates how many nets were located. The status area box lists all the net instances located.
5. If you want to zoom in on a search result, select *Zoom to Object*, otherwise, go to step 6.
6. To view a search result, select a net instance in the status area box.

   When you select a result to view, the page containing the object appears with the object highlighted.
7. Select another net in your design or click *Close* to navigate to a different net.

Running Scripts

You can create a text file containing a list of Concept HDL commands (a script) to run in batch mode. Scripts can call other scripts and can be interactive.

Within a script

- The `pause` command temporarily interrupts the Concept HDL editor until you press a key.
- The `echo` command displays messages from the script file in the Concept HDL console window. This lets you track the progress of a script and is useful for debugging.
- You can specify X-Y coordinates in a script.
- You can use environment variables in a script using the `($ENV_VARIABLE)` syntax.

  For example, if you are maintaining all your scripts at `/net/foo/script_home` and want to call a script named `check.scr` located at `/net/foo/script_home` from within your script, you can set the environment variable `$SCRIPT_HOME=/net/foo/script_home` and use the following command in your script file:

  ```
  script ($SCRIPT_HOME)/check.scr
  ```
- You can include user input tokens to allow a script to request user inputs during an operation. For more information, see User Input Tokens on page 347.

User Input Tokens

User input tokens must be placed at the beginning of a new line. There are two input tokens:

$<	When Concept HDL encounters this token in a script, it prints from the token to the end of the text line as a prompt, then waits for one item of input. The input can be a typed line, a pressed function key, a mouse point, or a *Ctrl + C* operation. You cannot press *Enter* in response to a user input request.
$;	This token also prints from the token to the end of the text line as a prompt and waits for input, but this token accepts and interprets input until you enter a semicolon. If this token is included, Concept HDL follows the prompt with the message: Type `;` when done with user input.

Running a Script

You can run a script in the following four ways:

- Running a script from the Tools menu on page 348
- Running a script from the Concept HDL console window on page 349
- Running a script from the Concept HDL console window on page 349
- Running a script from the UNIX or Windows command prompt on page 349

Running a script from the Tools menu

1. Choose *Tools* > *Script*.

   The *Open* dialog box appears.
2. Navigate to the script file you want to run and highlight the filename.
3. Click *Open*.

Running a script everytime you open a project in Concept HDL

If you want to automatically run a script everytime you open a project in Concept HDL, do the following:

1. Choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.
2. Select the *Paths* tab.
3. Enter the name of the script file in the *Input Script* field or click *Browse* to select the script file.

If you want to run more than one script everytime you open a project in Concept HDL, create a master script file and list the sequence in which you want to run the scripts in the master script file. Specify the name of the master script file in the *Input Script* field. For example, if you want to run a script named `check.scr` and then run a script named `zoom.scr`, do the following:

1. Create a script file named `master.scr` with the following entries:

```
script check.scr
script zoom.scr
```

2. Enter `master.scr` in the *Input Script* field in the *Paths* tab of the *Concept Options* dialog box.

Running a script from the Concept HDL console window

➤ Use the `script <file_name>` command

You can specify the path to the script file or use an environment variable to specify the path to the script file. For example, if you are maintaining all your scripts at `/net/foo/script_home` and want to run a script named `check.scr` located at `/net/foo/script_home` from the console window, you can set the environment variable `$SCRIPT_HOME=/net/foo/script_home` and use the following command in your script file:

```
script ($SCRIPT_HOME)/check.scr
```

For more information on the `script` command, see Script on page 583.

Running a script from the UNIX or Windows command prompt

➤ Use the following command:

```
nconcepthdl -proj <project_name>.cpm -scr <script_file_name>
```

For more information on the `nconcepthdl` command, see Nongraphical Concept HDL (nconcepthdl) on page 611.

Stopping a Script

➤ Use *Ctrl+C* to stop a script.

Sample Scripts

Some simple examples of scripts are given below.

Script to add a LS04 component to a drawing and use the mouse to position the part

```
add ls04
$<Place the LS04
```

Script to run multiple script files in a specific sequence

```
script /net/foo/scripts/set_options.scr
script /net/foo/scripts/check.scr
```

```
script /net/foo/scripts/zoom.scr
```

Script to add a SIZE property to a part with a size specified at the time of entry

```
property
$<Choose the part to add a size to
size =
$<Type in the size you want and press Enter
$<Place the property on the drawing
```

Script to rotate an object until the user enters a semicolon

```
rotate
$;Rotate the object until properly oriented
```

A more complicated script might contain a large number of `signame` commands and prompt the user for a point to place each `SIG_NAME` property.

Highlighting (Cross-Probing) Objects

You can highlight selected objects

- In expanded drawings - to trace a signal on multiple pages of a drawing and across multiple levels.
- Between Concept HDL and other system tools - to correlate the circuit logic to changes you made in the schematic or to navigate nets between a physical layout and the corresponding schematic.

For more information on highlighting and dehighlighting objects, see Highlighting Objects on page 105 and Turning Off Highlighting on page 105.

Distributing Design Changes between Physical and Logical Designs

1. Choose *Tools > Design Differences*.

   The *Design Differences* dialog box appears.

2. Select one or both options to:
   - update the board view to specify the Allegro board name in the Allegro Board box.
   - update the package view.

3. Click *OK*.

   See the Design Synchronization online help for more information on handling design differences.

Applying Connectivity Changes from the Physical Design to Your Schematic

1. Choose *Tools > Design Association*.

   The *Design Association* dialog box appears with markers information.

2. Use the menu commands in the *Design Association* dialog box to apply the connectivity changes.

   See the *Design Association online help* for more information on applying connectivity changes from the board layout to your schematic.

Back Annotating Your Design

➤ Choose *Tools > Back Annotate*.

Concept HDL reads the `pstback.dat` file containing the physical part and adds the information to the design.

Module Ordering

In hierarchical designs, you can change the order in which child blocks are plotted and cross-referenced. You can also exclude certain modules from being cross-referenced or plotted. Concept HDL lets you reorder modules using drag and drop operations. You can exclude or include modules by simply right clicking on them and selecting a menu item.

Note: In the context of module ordering, a module refers to a hierarchical block that has a schematic associated with it.

The following conditions apply to module ordering:

- The modules that are being re-ordered must be at the same level of hierarchy.
- The modules that are being re-ordered must have the same parent module.

Module ordering saves all ordering and the exclusion or inclusion information in a module order file named `module_order.dat` in the <*root design*>/`sch_1` directory. The `module_order.dat` file is read during cross referencing and hierarchical plotting.

The hierarchy of modules is displayed in the form of a tree. You can choose to view the tree with or without the modules that have been excluded.

Changes in Module Ordering from Concept HDL 14.0 to Concept HDL 14.2

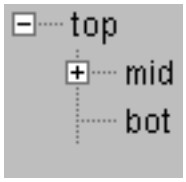
In Concept HDL 14.0, if there are multiple occurrences of a module in a design, the *Module Ordering* dialog box displays only one occurrence of the module in Hierarchy or Expanded mode and all occurrences of the module in Occurrence Edit mode. In Concept HDL 14.2, the *Module Ordering* dialog box displays all occurrences of the module in Hierarchy, Expanded and Occurrence Edit mode.

Important

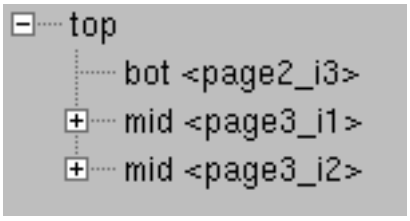
If you open a design created in Concept HDL 14.0 in Concept HDL 14.2 and perform module ordering or hierarchical plotting (in Windows or HPF Plotting), the `module_order.dat` file (located at <`worklib`>/<`root design`/`sch_1`) is written in a new format. If you open the design again in Concept HDL 14.0, you cannot see the module ordering information in Concept HDL 14.0. Cadence recommends that you maintain a copy of the `module_order.dat` file created by Concept HDL 14.0 or maintain a copy of the design so that you do not lose the module ordering information in Concept HDL 14.0.

In Concept HDL 14.0, if you have done module ordering in both Hierarchy mode and Occurrence Edit mode, you will lose the module ordering that you did in Hierarchy mode when you move to Concept HDL 14.2. In Concept HDL 14.2, if you open the *Module Ordering* dialog box in Hierarchy mode, the *Module Ordering* dialog box will display the module ordering information that was displayed in the Occurrence Edit mode in Concept HDL 14.0. For example, suppose that the module ordering information in Concept HDL 14.0 was as below:

Module Order in Hierarchy Mode

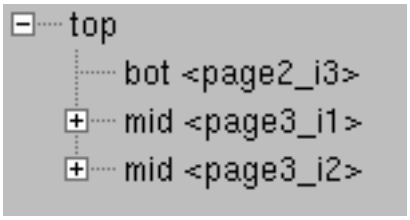


Module Order in Occurrence Edit Mode



There are two occurences of `mid` in the design.

When you open the design in Concept HDL 14.2 and open the *Module Ordering* dialog box in Hierarchy mode, the module order will be displayed as:



Concept HDL 14.2 displays all occurrences of a module in Hierarchy mode also.

Note: In Concept HDL 14.0, if you have done module ordering only in Hierarchy mode or only in Occurrence Edit mode, you will not lose any module ordering information in Concept HDL 14.2.

Using Module Ordering

You can change the order of design modules through the *Edit > Module Order* menu. The *Module Ordering* dialog box appears where you can perform the ordering and exclusion or inclusion operations.

Reordering of Modules

To change the order of a module, drag it and drop it on the module after which you want to insert it. If you want to make the module the first child of its parent, drop the module on the parent.

Note: If you perform page renumbering, the module order will not change automatically. You must manually change the order of the modules in the *Module Ordering* dialog box. For more information, see Page Renumbering, Module Ordering and Hierarchical Plotting on page 368.

Exclusion of a Module

Module ordering allws you to selectively exclude module(s) from the root module. Excluding a module will automatically exclude everything inside that module.

You can exclude modules that are not required to be plotted or cross-referenced to avoid cluttering of the module with the same reusable module. You cannot exclude the root module.

A module can be excluded in two ways:

- Project-Specific Exclusion
- Global Exclusion

Project-Specific Exclusion

Project-specific exclusion applies to the currently opened project. A module can be excluded from the root module in the following way:

1. Right click on the module in the tree to invoke a context menu.
2. If the module is not already excluded, the menu will have two options enabled:
   - Exclude Occurrence

     Excludes only the current occurrence of the module
   - Exclude All

     Excludes all occurrences of the module

Global Exclusion

Global exclusion is used when modules are required to be excluded from the hierarchy at the site level. This is achieved through the file `xmodules.dat` in <*`your_install_dir`*>/`share/cdssetup`.

The `xmodules.dat` file is initially empty and can be changed at the client site. You can also keep this file in $*`HOME`*/`cdssetup` or <*`proj_dir`*>/`cdssetup` to customize it to your own needs. It consists of a listing of module names only. The modules listed in this file are treated as any other excluded module.

For more information about this file, see Excluded Modules File on page 357.

When the hierarchy tree is constructed, with no existing module order file, the `xmodules.dat` file is searched using the CSF mechanism and the modules found in this file

are excluded from the hierarchy tree. For more information on CSF mechanism, see CSF Mechanism on page 355.

If the contents of the `xmodules.dat` file are changed during a Concept HDL session, these changes are reflected in the *Module Ordering* dialog box. If the user clicks on the *Reset* button, all the ordering information is lost.

Note: The modules excluded in the `xmodules.dat` file not be included if you click the *Reset* button.

CSF Mechanism

The `xmodules.dat` file is searched for in directories in the order specified in the `setup.loc` file, which is located in the <*your_install_dir*>/share/cdssetup directory.

An example of the `setup.loc` file is:

```
cwd should always be searched first
$CDS_WORKAREA           user workarea if defined
$CDS_SEARCHDIR          this is set by various tools during tool startup
$HOME
$CDS_PROJECT            project storage area, ignored if not defined
$CDS_SITE               Site Setup Info - default is $CDS_INST_DIR/share/local
$CDS_INST_DIR/share     Cadence Default Setup Info
```

If an environment variable is not defined, its entry is ignored and no errors are generated.

If the paths specified in the `setup.loc` file contain the `cdssetup` directory, the `xmodules.dat` file is searched for in the `cdssetup` directory too.

Inclusion of a Module

An excluded module can be included in the hierarchy in the following way:

1. Right click on the excluded module to invoke a context menu.
2. The context menu will have two options:
   - Include Occurrence

     Includes only the current occurrence of the module
   - Include All

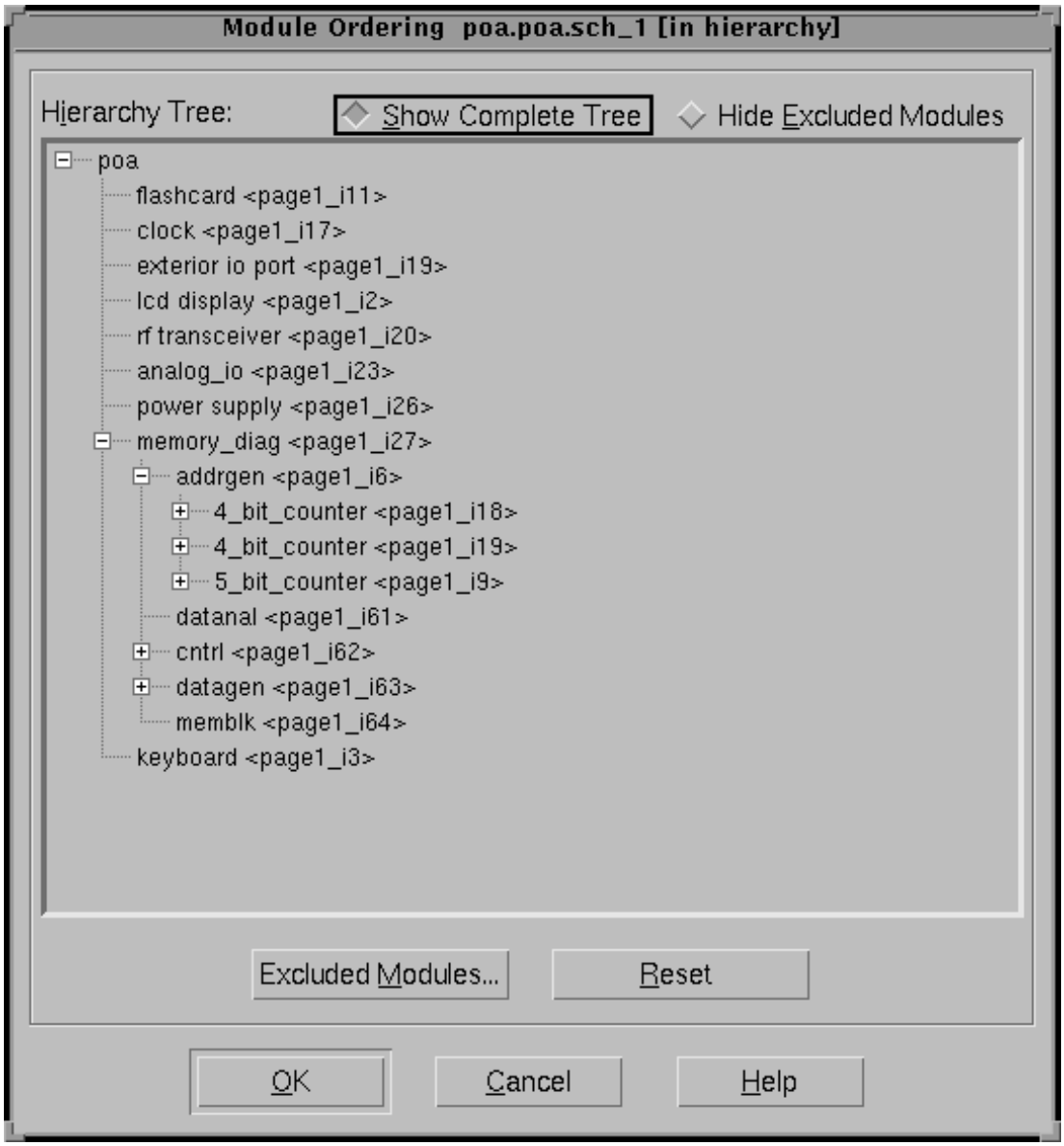
     Includes all occurrences of the module

Note: A module excluded by specifying it in the `xmodules.dat` file cannot be included again using the *Module Ordering* dialog box.

For module ordering, perform the following steps:

1. Choose *Edit > Module Order*.

   The *Module Ordering* dialog box appears.



   The modules under the root design and each occurrence of a module in the hierarchy are displayed. For example, in the above figure, two instances of the `4_bit_counter` module are displayed under the `addrgen` module. This means that the module `4_bit_counter` is instantiated twice in `addrgen` module.

2. To select the modules to view, choose *Show Complete Tree* or *Hide Excluded Modules*.

   - If you choose *Show Complete Tree*, you can see the complete hierarchy of modules, including the modules that have been excluded.
   - If you choose *Hide Excluded Modules*, the excluded modules are hidden.

3. To exclude a module, right click on that module. A context menu appears. Choose *Exclude All* or *Exclude Occurrence*.

   - If you choose *Exclude All*, Concept HDL excludes all occurrences of the module.

     You can also use the `xmodules.dat` file to exclude all occurrences of a module. For more information, see Global Exclusion on page 354.

   - If you choose *Exclude Occurrence*, Concept HDL excludes the current occurrence of the module.

4. To include an excluded module, right click on that module. A context menu appears. Choose *Include All* or *Include Occurrence*.

   - If you choose Include All, Concept HDL includes all occurrences of the module.
   - If you choose Include Occurrence, Concept HDL includes the current occurrence of the module.

5. Click *Excluded Modules* to view a list of all excluded modules.

   If a module has multiple occurrences, and all its occurrences have been excluded using *Exclude All*, this dialog shows all the excluded occurrences.

6. Click *Reset* to clear all the exclusions and inclusions made. All the ordering information is lost, and the default state of the hierarchy of modules is restored.

7. Click *OK*.

   Concept HDL saves the module ordering information in the `module_order.dat` file.

Excluded Modules File

Apart from excluding modules through the *Module Ordering* dialog box, modules can also be excluded by mentioning the module name in a file named `xmodules.dat`. This file can reside in the following places:

- Hierarchy - <*your_inst_dir*>/share/cdssetup
- Home - *$HOME*/cdssetup

- Project - <*proj_dir*>/cdssetup

Note: For a given project, the file at the project level is given precedence over other projects. If the file is not present in the <*proj_dir*>/cdssetup then the one in the Home directory takes precedence over the project in hierarchy and applies to all of the projects. If there is no such file in the Home directory, the information is read from the one in the hierarchy and it applies to all projects at the site.

The format of the xmodules.dat file is

```
(“<module-name-1>” “<module-name-2>” “<module-name-3>”)
```

For example, if a user has two hierarchical blocks, capacitor and gnd, and wants to exclude them from plotting and cross referencing, the xmodules.dat file should appear as

```
(“capacitor” “gnd”)
```

Important

Ensure that there are no spaces before or after the module name. For example, to exclude a module named clock, if you specify (“ clock”) in the xmodules.dat file, the module will not be excluded. Note that you can have spaces within the module name. For example, to exclude a module named power supply, you can specify (“power supply”) in the xmodules.dat file.

Note: If a module is manually removed from the xmodules.dat file, the module_order.dat file is not updated. The module still appears excluded in the *Module Ordering* and the *Plot* dialog boxes. If you want to include this module again, either reset the module_order.dat file or include the module in the *Module Ordering* dialog box.

Displaying and Working with Schematic Page Numbers

Concept HDL allows you to display page numbers on a schematic page using custom text variables for page numbers. You can modify the custom text to change the way in which page numbers are displayed. You can also renumber the pages in a design. For more information on displaying and working with schematic page numbers, see the following sections:

- Displaying Page Numbers in a Schematic on page 359
- Modifying Custom Text for Page Numbers on page 362
- Updating Custom Text Variables for Page Numbers on page 363
- Renumbering Schematic Pages on page 364

Displaying Page Numbers in a Schematic

Concept HDL allows you to display page numbers on a schematic page using custom text variables for page numbers. When you plot a schematic page, the page number of the schematic page is plotted only if you have added the custom text variables on the schematic page.

Cadence recommends that you add custom text variables for page numbers on the schematic pages. This allows you to easily refer to a page in Concept HDL against a plotted page or a cross-reference report using the *File > Edit Page/Symbol > Go To* command. For more information, see How do I go to a specific page in a design? on page 59.

- If you plot a schematic page on which custom text variables for page numbers are not added, the plotted page will not contain the page number. If you later want to refer to a page in Concept HDL against its plotted page, you will not know which page to open in Concept HDL.
- When you perform cross-referencing on a design, the cross-reference reports display the schematic page numbers. It will be easier to refer to a page in Concept HDL with the cross-reference report if the page number is displayed on the schematic page.

  Note: The cross-reference reports will contain the page number information even if you have not added the custom text variables on the schematic page.

The following custom text variables allow you to display page numbers on a schematic page. For more information on custom text variables, see Working with Custom Text on page 239.

- CON_PAGE_NUM
- CON_TOTAL_PAGES
- CURRENT_DESIGN_SHEET
- TOTAL_DESIGN_SHEETS

The CON_PAGE_NUM and CON_TOTAL_PAGES Custom Text Variables

The `CON_PAGE_NUM` and `CON_TOTAL_PAGES` variables allow you to display the page numbering information for pages in a cell. For example, if the root design `A` in a hierarchical design has three pages, and sub design `B` has four pages, the value of:

- `CON_PAGE_NUM` variable will be 1 for the first page of the root design `A` and 3 for the last page in the root design `A`. The value of the `CON_PAGE_NUM` variable for the first page of sub design `B` will also be 1 and the value of the variable for the last page of sub design `B` will be 4. If you are in the third page of the sub design `B`, the value of the

CON_PAGE_NUM will be 3 and not 6. This means that the CON_PAGE_NUM variable does not take into account the pages in all the cells in a hierarchical design.

- CON_TOTAL_PAGES variable will be 3 for the root design A and 4 for the sub design B. This means that the CON_TOTAL_PAGES variable does not take into account the pages in all the cells in a hierarchical design.

  The value of CON_TOTAL_PAGES will be the value of the highest page number you have assigned for a page in the cell. For example, if you have four pages in a cell with numbers 1, 2, 3, and 6, the value of CON_TOTAL_PAGES is 6.

The CURRENT_DESIGN_SHEET and TOTAL_DESIGN_SHEETS Custom Text Variables

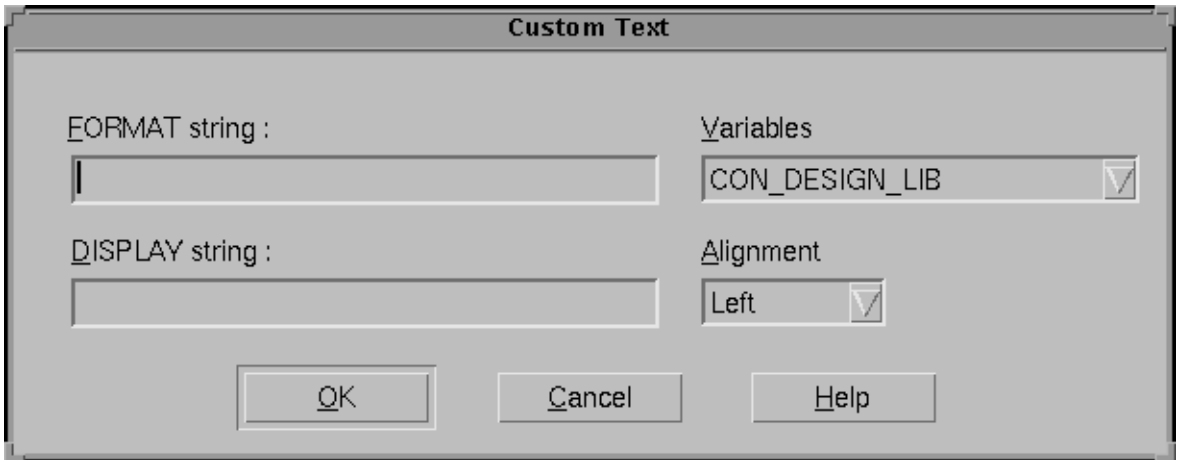
The CURRENT_DESIGN_SHEET and TOTAL_DESIGN_SHEETS variables allow you to display the page numbering information for pages in a hierarchical design. For example, if the root design A in a hierarchical design has three pages, and sub design B has four pages, the value of:

- CURRENT_DESIGN_SHEET variable will be 1 for the first page of the root design A and 3 for the last page in the root design A. The value of the CURRENT_DESIGN_SHEET variable for the first page of sub design B will be 4 and the value of the variable for the last page of sub design B will be 7.
- TOTAL_DESIGN_SHEETS variable will be 7. The TOTAL_DESIGN_SHEETS variable takes into account the actual number of pages in a hierarchical design. Even if the highest page number assigned to a page in a hierarchical design is greater than the actual number of pages in the design, the TOTAL_DESIGN_SHEETS variable will display only the actual number of pages in the design.

To add the custom text variables for page numbers

1. Choose *Text > Custom Text*.

The *Custom Text* dialog box appears.



2. Select the following variables from the *Variables* drop-down list as required:

- `CON_PAGE_NUM`
- `CON_TOTAL_PAGES`
- `CURRENT_DESIGN_SHEET`
- `TOTAL_DESIGN_SHEETS`

The variable is displayed in the *FORMAT string* field. The *Display string* field displays the current value of the variable. For example, if you are adding the `CURRENT_DESIGN_SHEET` variable on the 15th page of a hierarchical design, the *Display string* field displays the value `15`.

3. Edit the format string for the custom text as required.

For example, if you have selected the `CURRENT_DESIGN_SHEET` the *FORMAT string* field displays `<CURRENT_DESIGN_SHEET>` and the *Display string* field displays, say `15`. If you want to display the page number as `PAGE 15` on the schematic page, change the text in the *FORMAT string* field to `Page <CURRENT_DESIGN_SHEET>`. The *Display string* field now displays `PAGE 15`.

Similarly, if you have selected both the `CURRENT_DESIGN_SHEET` and `TOTAL_DESIGN_SHEETS` variables, and want to display running-total page numbers, such as `Page 15 of 20`, change the text in the *FORMAT string* field to:

```
Page <CURRENT_DESIGN_SHEET> of <TOTAL_DESIGN_SHEETS>
```

The *Display string* field displays:

```
PAGE 15 of 20
```

Note: If you want to add an environment variable to the format string, precede it with a

$ sign. The current value of the environment variable is displayed in the *Display string* field.

4. Click *OK*.

   The custom text for the page number gets attached to the cursor.

5. Click on an object to attach the custom text to it.

   Note: You can add multiple custom text to the same object on the schematic.

6. Click again to place the custom text on the schematic.

   The page number is displayed in the schematic.

7. Right click and choose *Done*.

Note: The operations like delete, move, copy, rotate and spin that can be performed on properties can be done on the custom text also.

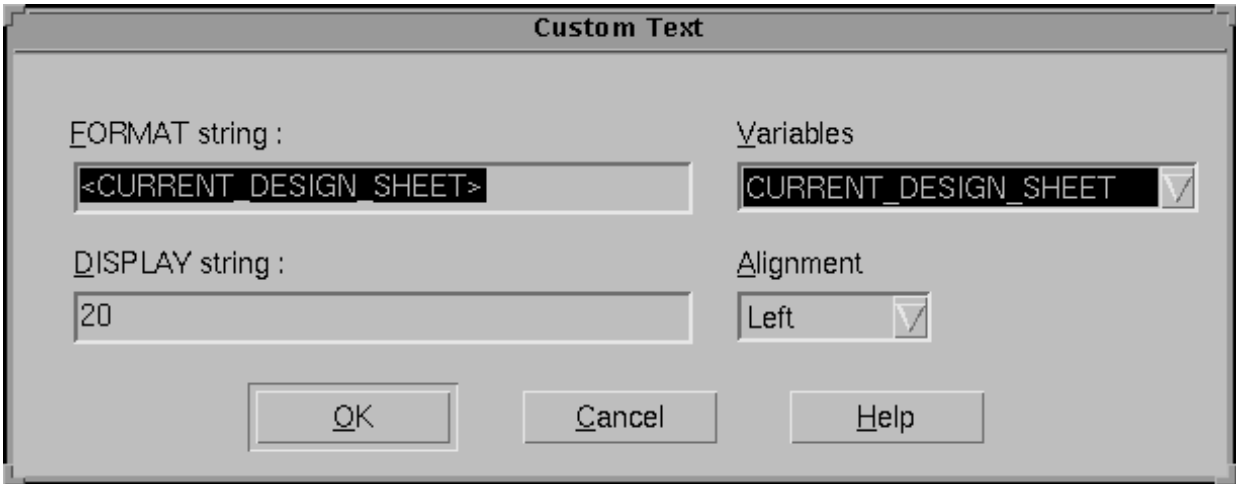
Modifying Custom Text for Page Numbers

Concept HDL allows you to modify the custom text variables for page numbers to change the way in which the page numbers are displayed.

For example, if you have added only the `CURRENT_DESIGN_SHEET` custom text variable in the 20th page in a hierarchical design that has 25 pages, the schematic page will display the page number as `20`. Now if you want to display running-total page numbers such as `Page 20 of 25`, do the following:

1. Choose *Text > Change*.
2. Click on the page number custom text `20` that is displayed on the schematic page.

The *Custom Text* dialog box appears.



3. Select the `TOTAL_DESIGN_SHEETS` variable from the *Variables* drop-down list.
4. Modify the text in the *FORMAT string* field to:

   ```
   Page <CURRENT_DESIGN_SHEET> of <TOTAL_DESIGN_SHEETS>
   ```

   The *Display string* field displays:

   ```
   PAGE 15 of 20
   ```
5. Click *OK*.

   The page numbering information on the schematic changes to `PAGE 15 of 20`.

Note: The page number format is always integer. Concept HDL does not allow you to change the page number format to `i`, `I`, `a`, etc.

Updating Custom Text Variables for Page Numbers

The `CON_PAGE_NUM` and `CON_TOTAL_PAGES` custom text variables for page numbers are automatically updated whenever any changes are made in the design. However, you may need to update the `CURRENT_DESIGN_SHEET` and `TOTAL_DESIGN_SHEETS` custom text variables for page numbers to ensure that the schematic page displays the correct page number. This update needs to be done in the following cases:

- If you have modified the design by adding or deleting pages or blocks.
- When you add custom text variables for page numbers on a schematic page, the name of the variable is substituted by the page number. If the schematic page continues to display the variable name instead of the page number, you need to update the variable. For example, if you add the `CURRENT_DESIGN_SHEET` custom text variable in the 20th

page in a hierarchical design, the schematic page may display <CURRENT_DESIGN_SHEET> instead of 20.

- When you renumber schematic pages using the page renumbering commands, the schematic pages may not display the correct page number. For more information on renumbering pages, see Renumbering Schematic Pages on page 364.

Note: When you perform module ordering, cross-referencing, or plotting of the design, the custom text variables are updated automatically.

To update the custom text variables for page numbers

➤ Choose *Text > Update Sheet Variables*.

The custom text variables for page numbers on all pages in the design are updated to display the correct page number.

You can also use the `updatesheetvars` console command to update the custom text variables for page numbers. For more information, see Updatesheetvars on page 603.

Renumbering Schematic Pages

In multiple page designs, you may have non-contiguous pages in the design. You may want to remove those pages so as to make your design contiguous. There can also be situations when you would want to add a new page in the middle of a design.You can use the page renumbering feature of Concept HDL to perform these tasks. Page renumbering lets you change the order in which pages are arranged in the design. You can interchange two pages, move a page from one place to another, or delete a page. Concept HDL lets you renumber pages through a number of console commands. For more information on page renumbering commands, see Page Renumbering Commands on page 366.

In the context of page renumbering, there are two types of page numbers, physical and logical.

Physical Page Numbers

Physical page numbers are the ones that you see on the Concept HDL title bar. For example, if `CLOCK.SCH.1.4` is displayed on the Concept HDL title bar, `CLOCK` indicates the name of the cell, `SCH` indicates that the view type is schematic, 1 indicates the version number of the view and 4 indicates the physical page number of the schematic page.

The physical page numbers change when you renumber schematic pages.

Logical Page Numbers

When you create a new schematic page, the logical and physical page numbers are the same. Concept HDL assigns the logical page number for a schematic page using the directive `set page_number` *Pn* to the `page*` files. The logical page number is the original page number for the schematic page. When you renumber a schematic page, the physical page number changes but the logical page number does not change. Concept HDL keeps track of the logical page numbers of pages that have been renumbered by reading the `set page_number` *Pn* in the `page*` files.

For example, when you create a three page design, the logical and physical page numbers of the schematic pages will be 1, 2 and 3. While cross-probing or globally locating objects, canonical names are shown with the logical page numbers. For example, the canonical name `@top_lib.top.(sch_1):page2_i5` displayed in the *Global Find* dialog box means that the component that has the `PATH=i5` property is located in the logical page 2 in the `sch_1` view of the cell `top` in the library named `top_lib`.

In the above design, if you swap page 2 with page 3, and perform a global find for the component that originally existed on page 2 of the design, but now exists on page 3 of the design, the *Global Find* dialog box continues to display the canonical name for the component as `@top_lib.top.(sch_1):page2_i5`. This means that the logical page number has not changed even after you swapped the pages in the design. If you click on the canonical name in the *Global Find* dialog box, Concept HDL zooms in to display the object on page 3 (the physical page) of the design.

Using Page Renumbering

If you are renumbering the pages of a design that is not of the current release, you need to first save the design using the `hier_write` console command. Writing the design once adds the directive `set page_number` *Pn* to the `page*` files. The number *Pn* is called the logical page number. This directive is required for Concept HDL to keep track of the original page numbers of pages that have been renumbered.

If you have renumbered the pages of a design but have not changed the schematic, you do not need to save the design. The change is visible in all the tools.

An example of the need to use the page renumbering feature would be a design that has 12 pages of which page numbers 5 and 6 are blank. You may want to remove the blank pages and collapse the design. To do this, you can move page 7 to 5, page 8 to 6, page 9 to 7, and so on.

You may need to add a new page somewhere in the middle of a design. You can use page renumbering for this. You can move the pages back, and then you would have a page on which you can work. For example, if you have pages 1 to 10 and you want to add another

page in between and number it 8, you have to move page 10 to 11, page 9 to 10, page 8 to 9. Then, go to page 8.

Page Renumbering Commands

You can renumber the pages of a design through the following console commands:

- page move
- page swap
- page delete
- page reset
- page forcereset

You cannot run page renumbering commands in a design if a page in the same design is opened by another user who has write permissions. Concept HDL displays the following error message in the console window if you run page renumbering commands when the design is opened by another user:

```
This design is being simultaneously edited by multiple users. Ignoring page command.
```

Note: If you are working on a block used in the design and another user is working on another block, you can run page renumbering commands on the pages in the block.

page move

```
page move <@lib.cell(view)> X Y
```

Moves existing page X to a non-existing page Y.

Note: You cannot move a page to a page that already exists.

page swap

```
page swap <@lib.cell(view)> X Y
```

Swaps existing pages X and Y.

page delete

```
page delete <@lib.cell(view)> X
```

Deletes an existing page X.

Note: The move, swap, and delete commands cannot be executed on pages if:

- the page is currently being edited.
- the page is being edited, and changes in the page have not been saved.

Note: You need not save or repackage the design after you run the move or swap command. This is because the canonical names in the design do not change after you run the move and swap commands. If you run the delete command, you must save and repackage the design.

page reset

```
page reset X
```

Sets the existing page number to X.

This command sets the logical page number of the currently open page to X. X is a logical page number. When this command is run, Concept HDL first checks the existence of the logical page number X in the physical page files. This command is executed only if the logical page X does not exist in any of the physical page files.

page forcereset

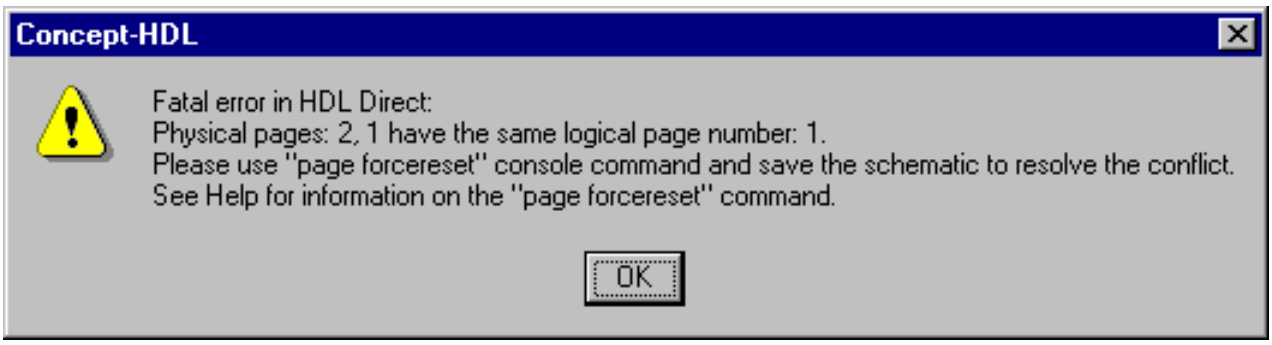
```
page forcereset X
```

This command sets the logical page number of the currently open page to X. This command is the same as the `page reset` command except for the fact that Concept HDL does not check for the existence of the logical page number X in any of the physical page files before setting the logical page number X to the currently open page.

You should keep the following points in mind while executing the `page reset` and `page forcereset` commands:

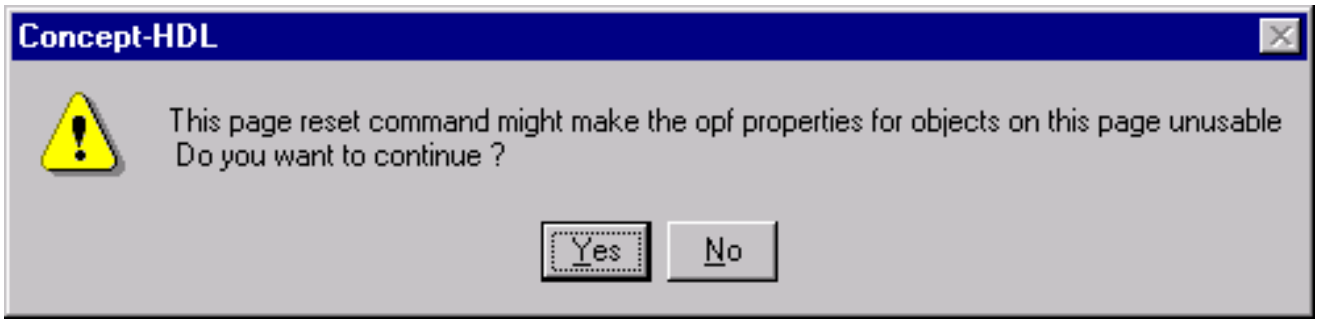
1. To commit the change made by the `page forcereset` or `page reset` command, you have to save the page.To reflect this status, a * is shown on the Concept HDL title bar. If you want to de-commit the change made by any of these commands, do not save the page because you cannot undo the command.

2. If two different physical pages have the same logical page number, and you try to save the design, Concept HDL produces the following error message:


Concept-HDL
Fatal error in HDL Direct:
Physical pages: 2, 1 have the same logical page number: 1.
Please use "page forcereset" console command and save the schematic to resolve the conflict.
See Help for information on the "page forcereset" command.
OK

   You can then use the `page forcereset` command to change the logical page numbers.

3. Running the `page forcereset` and `page reset` commands makes the occurrence properties of objects, which are on renumbered pages, unusable. You get the following error message:


Concept-HDL
This page reset command might make the opf properties for objects on this page unusable
Do you want to continue ?
Yes
No

4. While cross-probing or globally locating objects, canonical names are shown with the logical page numbers. For example, the canonical name `@top_lib.top.(sch_1):page2_i5` displayed in the *Global Find* dialog box means that the component that has the `PATH=i5` property is located in the logical page 2 in the `sch_1` view of the cell `top` in the library named `top_lib`.

Page Renumbering, Module Ordering and Hierarchical Plotting

When you do page renumbering, the order of the pages in the design hierarchy tree in the *Module Ordering* dialog box and the *Plot* dialog box does not change automatically. You must manually change the order of the pages in the *Module Ordering* dialog box. For example, suppose that you have a block `POA` with:

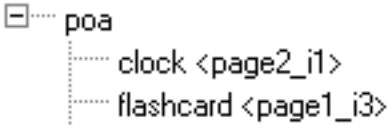
- Block `clock` instantiated on the page 1 and
- Block `flashcard` instantiated on page 2.

The *Module Ordering* dialog box displays the following design hierarchy tree:

The *Plot* dialog box displays the following design hierarchy tree:

Now swap page 1 with page 2 using the `page swap 1 2` Concept HDL console command.

Hierarchy Tree in Module Ordering Dialog Box after Page Swap



Note that though the page numbers in the hierarchy tree in the *Module Ordering* dialog box have changed, the order of the pages has not changed. You must manually change the order of the pages in the *Module Ordering* dialog box, as shown in Figure 14-1 on page 370.

Hierarchy Tree in Plot Dialog Box after Page Swap

Note that the page number for the block `clock` has not changed from 3 to 4 and the page number for the block `flashcard` has not changed from 4 to 3 in the *Plot* dialog box. This results in the pages for the blocks not being plotted in the expected order. In this example, you would expect the pages in the design to be plotted in the following order:

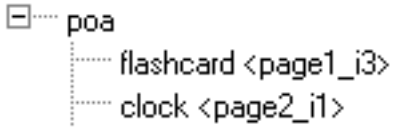
1. Page 1 of block `POA` that has the block `clock` instantiated on it
2. Page 1 of block `POA` that has the block `flashcard` instantiated on it
3. Pages of block `clock`
4. Pages of block `flashcard`

However, the pages are plotted in the following order:

1. Page 1 of block `POA` that has the block `clock` instantiated on it
2. Page 1 of block `POA` that has the block `flashcard` instantiated on it
3. Pages of block `clock`
4. Pages of block `flashcard`

If you want the pages for the block `flashcard` to be plotted before the pages for the block `clock`, you must change the order of the pages in the *Module Ordering* dialog box as shown in the figure below:

Figure 14-1 Change in Hierarchy Tree in Module Ordering Dialog Box



15

Netlisting Your Design

A netlist contains the list of signals, parts, and pins in a design and information on how they are connected with other devices.

Concept HDL allows you to generate netlists that can be used for:

- Packaging the design.

  For more information, see Netlisting for Packaging the Design on page 372.

- Digital simulation

  For more information, see Netlisting for Digital Simulation on page 376.

- Programmable IC simulation

  For more information, see Netlisting for Synthesizing a Design in Synplify on page 378.

- Analog and Mixed Signal simulation

  For more information, see Netlisting for Analog and Mixed Signal Simulation on page 382.

You can specify the options for generating the netlist.

When Concept HDL generates the netlist, it does the following:

- Checks the drawings for Verilog and VHDL compatibility.
- Performs cross-view checking of ports, port modes, and port types between schematic views and symbol views. For more information, see Cross-View Checking on page 633.
- Performs entity declaration checking for instantiated components.

  For more information, see Entity Declaration Checking for Instantiated Components on page 636.

See Netlisting Errors on page 636 for solutions or workarounds for errors that occur during the netlisting process.

Netlisting for Packaging the Design

Concept HDL generates the Verilog netlist for used for packaging the design when you save the design. You can specify the options for generating the netlist for packaging the design.

To specify the options for generating the Verilog netlist used for packaging the design

1. In Concept HDL, choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab.

   Ensure that the *Create Netlist* check box is selected.

   Verilog netlisting is enabled by default. You can select the *VHDL* check box if you want to generate the VHDL netlist for the design.

3. Click the *Options* button next to the *Verilog* check box.

   The *Verilog Netlist* dialog box appears.

4. Select the *Verbose Output* check box to log the debug messages of the netlisting process in the `hdldir.log` file located in the `<project_directory>/temp/` directory.

5. Select the *Analyze on Save* check box if you want the netlist to be analyzed by `ncvlog.exe` (NC Verilog compiler) every time you save the design in Concept HDL.

   This check box is enabled only if you have selected the NC Verilog simulator in the *Tools* tab of the *Project Setup* dialog box.

6. Select the *Check Instance vs Signal* check box if you want Concept HDL to check if the name of any signal on the schematic is the same as `page<page_number>_<value of PATH property on any instance>`. If this check box is selected, Concept HDL displays the following error message for every signal that has the same name as `page<page_number>_<value of PATH property on any instance>`:

   ```
   126 ERROR "Identifier is used as both a PATH value and a signal name."
   ```

7. Specify the maximum number of netlisting errors that you want to allow in the *Max Errors* field. If the number of netlisting errors in the design exceeds the number specified here, Concept HDL will not generate the netlist.

8. Specify the timescale directive for the Verilog module of the schematic in the *Time Scale* field. The default value is `1ns/1ns`.

9. Specify the Verilog logic type for all nets in the design. You can use any legal Verilog net type, such as `WIRE`, `WAND`, and `WOR`. The default value is `WIRE`.

   Note: The specified net type applies to all drawings in the design. You can override the net type for individual drawings by using the `VLOG_NET_TYPE` property on a `VERILOG_DECS` symbol. For more information on specifying the Verilog logic type of ports and signals, see Setting the Verilog Logic Type for Ports and Signals on page 157.

10. Specify the signal names for the Verilog net type `Supply 0`.

11. Specify the signal names for the Verilog net type `Supply 1`.

12. Click *OK* to save the settings and close the *Verilog Netlist* dialog box.

13. Click *OK* to close the *Concept Options* dialog box.

To generate the Verilog netlist used for packaging the design

➤ In Concept HDL, choose *File > Save*.

The Verilog netlist file `verilog.v` is generated in the following directory:

`<project_directory>/worklib/<design_name>/sch_1/`

If you are working on a large design, saving the design in Concept HDL takes a long time because the netlist for the entire design is regenerated every time you save the design. You can avoid this by disabling netlisting of the design. For more information, see Disabling Netlisting of Designs on page 374. When you package your design, the Verilog netlist for packaging the design is generated even if netlisting is disabled. For more information see What Happens if Netlisting is Disabled and You Package the Design?.

The Verilog netlisting errors that are identified during the netlisting process are displayed in the *Markers* dialog box in Concept HDL. The VHDL netlisting errors are displayed only if the *VHDL* check box is selected in the *Output* tab of the *Concept Options* dialog box. Click on an error to highlight the area on the schematic where the error has occurred. Correct the schematic errors and generate the netlist again.

The schematic errors that are identified during the process of generating the netlist for packaging are stored in the `netlister.mkr` file that is located in the `<project_directory>/temp/xxnedtmp/` directory. You can view the errors by loading the file in the *Markers* dialog box. The `hdldir.log` file located in the `<project_directory>/temp/` directory contains the details of the netlisting process.

Note: The Verilog netlist for packaging the design is also regenerated when you package the design. During the packaging process only the Verilog netlisting errors, if any, are displayed. The VHDL netlisting errors, if any, are not displayed even if VHDL netlisting is enabled in the *Output* tab of the *Concept Options* dialog box. If there are any Verilog netlisting errors, the packaging process will stop. You have to correct the errors in the design and restart the packaging process. If there are any warnings in the design, the packaging process will not stop. The *Progress* dialog box that appears when the packaging process is run will display the warnings in the design and the packaging process will continue.

To view netlisting errors in the packaging netlist

1. In Concept HDL, choose *Tools* > *Markers*.

   The *Markers* dialog box appears.

2. Choose *File* > *Load* and open the `netlister.mkr` file located in the `<project_directory>/temp/xxnedtmp/` directory.

   The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box.

For more information on the errors displayed in the *Markers* dialog box, see Netlisting Errors on page 636.

Disabling Netlisting of Designs

If you are working on a large design, saving the design in Concept HDL takes a long time because the netlist for the entire design is regenerated every time you save the design. You can avoid this by disabling netlisting of the design.

To disable Verilog and VHDL netlisting

1. In Concept HDL, choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab.

3. Deselect the *Create Netlist* check box.

If both Verilog and VHDL netlisting are disabled the Verilog and VHDL netlisting errors will not be displayed in the *Markers* dialog box when you save the design.

Note: When you package your design, the Verilog netlist for packaging the design is generated even if netlisting is disabled. For more information see What Happens if Netlisting is Disabled and You Package the Design?.

To disable VHDL netlisting

1. In Concept HDL, choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab.

   Ensure that the *Create Netlist* check box is selected.

3. Deselect the *VHDL* check box.

If VHDL netlisting is disabled the VHDL netlisting errors will not be displayed in the *Markers* dialog box when you save the design.

What Happens if Netlisting is Disabled and You Package the Design?

When you package the design, the Verilog netlist for packaging the design is generated even if netlisting of the design is disabled. The Verilog netlisting errors, if any, are displayed in the *Markers* dialog box. If there are any Verilog netlisting errors, the packaging process will stop. You have to correct the errors in the design and restart the packaging process. If there are any warnings in the design, the packaging process will not stop. The *Progress* dialog box that appears when the packaging process is run will display the warnings and the packaging process will continue.

Note: If you want to package a design that has netlisting errors, select the *Continue expansion even if netlisting errors* check box in the *Expansion* tab of the *Project Setup* dialog box. If this check box is selected, the packaging process will continue even if the design has netlisting errors. For more information, see Chapter 5, “Project Creation and Setup.”

Note: When you package a design, only the Verilog netlisting errors, if any, are displayed. The VHDL netlisting errors in the design, if any, are not displayed.

Netlisting for Simulation

This section describes the following:

- Netlisting for Digital Simulation on page 376
- Netlisting for Synthesizing a Design in Synplify on page 378

■ Netlisting for Analog and Mixed Signal Simulation on page 382

Netlisting for Digital Simulation

The Concept HDL Digital Simulation Interface allows you to generate the Verilog and VHDL netlist for digital simulation. You can generate the netlist for use with the following simulators:

- Verilog-XL simulator
- Affirma NC Verilog simulator
- Third-party Verilog simulators
- Leapfrog VHDL simulator
- Affirma NC VHDL simulator
- Third-party VHDL simulators

You can specify the options for generating the netlist in the Concept HDL digital simulation interface.

To specify the options for generating the netlist for digital simulation

1. Start *Project Manager*.
2. Choose *File > Open*.
3. Select the project file (.cpm) and click *OK*.
4. Choose *Tools > Setup*.

   The *Project Setup* window appears.
5. Select the *Tools* tab.
6. Click *Simulation Setup*.

   The *Choose Simulator* dialog box appears.
7. Select the simulator you want to use for performing digital simulation
8. Click *Setup*.

   The setup dialog box for the simulator appears.
9. Select the *Netlist* tab and specify the options for generating the netlist.

Note: If you have a hierarchical design, you can select the *Single File Netlist* check box in the *Netlist* tab to generate a single file netlist for the entire design.

For more information on how to specify the options for generating the netlist for each of the simulators, see the *Concept HDL Digital Simulation User Guide*.

If you have already selected the simulator (steps 1 to 6 above) you want to use for performing digital simulation, you can also specify the netlisting options by doing the following:

1. In Concept HDL or Project Manager, choose *Tools* > *Simulate*.

   The simulation interface dialog box appears.

2. Click *Setup*.

   The setup dialog box for the simulator appears.

3. Select the *Netlist* tab and specify the options for generating the netlist.

   Note: If you want to generate a single file netlist for a hierarchical design, select the *Single File Netlist* check box in the *Netlist* tab.

For more information on specifying the netlisting options for each of the simulators, see the *Concept HDL Digital Simulation User Guide*.

To generate the netlist for digital simulation

Once you have specified the options for generating the netlist for a specific simulator, you can generate the netlist for use with the simulator.

To generate the netlist:

1. In Concept HDL or Project Manager, choose *Tools* > *Simulate*.

   The simulation interface dialog box appears.

2. Click *Run*.

   The *Simulation Progress Status* window appears displaying the progress of the netlisting and simulation processes. Click *Details* to view the details of these processes.

For the Verilog-XL, NC Verilog, and third-party Verilog simulators, the Verilog netlist file `verilog.v` is generated. For the Leapfrog, Affirma NC VHDL, and third-party VHDL simulators, the VHDL netlist file `vhdl.vhd` is generated. The netlist files are located in the following directory:

```
<project_directory>/worklib/<design_name>/sim_sch_1/
```

Note: If you have selected the *Single File Netlist* check box in the *Netlist* tab, a single file Verilog netlist `<design_name>.v` or a single file VHDL netlist `<design_name>.vhd` are generated (for hierarchical designs). These files are located in the *run* directory you specify in the simulation interface dialog box.

The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box in Concept HDL. Click on an error to highlight the area on the schematic where the error has occurred. Correct the schematic errors and generate the netlist again.

The schematic errors that are identified during the process of generating the netlist for digital simulation are stored in the `simNetlist.mkr` file that is located in the *run* directory you specify in the simulation interface dialog box. You can view the errors by loading the file in the *Markers* dialog box. The `netassembler.log` file located in the `run` directory contains the details of the netlisting process.

To view netlisting errors in the digital simulation netlist

1. In Concept HDL, choose *Tools* > *Markers*.

   The *Markers* dialog box appears.

2. Choose *File* > *Load* and open the `simNetlist.mkr` file located in the `run` directory you specified in the simulation interface dialog box.

   The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box.

Netlisting for Synthesizing a Design in Synplify

Concept HDL allows you to generate the Verilog and VHDL netlist that will be used to pass the entire design for synthesis to Synplify, the synthesis tool from Synplicity. You can specify the options for generating the netlist for synthesizing a design.

To specify the options for generating the Verilog netlist for synthesizing a design

1. In Concept HDL, choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Output* tab.

   Ensure that the *Create Netlist* check box is selected.

3. Click the *Options* button next to the *Verilog* check box.

The *Verilog Netlist* dialog box appears.

4. Select the *Verbose Output* check box to log the debug messages of the netlisting process in the `hdldir.log` file located in the `<project_directory>/temp/` directory.

5. Select the *Analyze on Save* check box if you want the netlist to be analyzed by `ncvhdl.exe` (NC VHDL compiler) or `cv.exe` (Leapfrog compiler) every time you save the design in Concept HDL.

   This check box is enabled only if you have selected the NC VHDL or Leapfrog simulator in the *Tools* tab of the *Project Setup* dialog box. If the NC VHDL simulator is selected, the netlist is analyzed by `ncvhdl.exe`. If the Leapfrog simulator is selected, the netlist is analyzed by `cv.exe`.

6. Select the *Check InstanceVs Signal* check box if you want Concept HDL to check if the name of any signal on the schematic is the same as `page<page_number>_<value of PATH property on any instance>`. If this check box is selected, Concept HDL displays the following error message for every signal that has the same name as `page<page_number>_<value of PATH property on any instance>`:

   ```
   126 ERROR "Identifier is used as both a PATH value and a signal name."
   ```

7. Specify the maximum number of netlisting errors that you want to allow in the *Max Errors* field. If the number of netlisting errors in the design exceed the number specified here, Concept HDL will not generate the netlist.

8. Specify the timescale directive for the Verilog module of the schematic in the *Time Scale* field. The default value is `1ns/1ns`.

9. Specify the logic type for all nets in the design. You can use any legal Verilog net type, such as `WIRE`, `WAND`, and `WOR`. The default value is `WIRE`.

   Note: The specified net type applies to all drawings in the design. You can override the net type for individual drawings by using the `VLOG_NET_TYPE` property on a `VERILOG_DECS` symbol.

10. Specify the signal names for the Verilog net type `Supply 0`.

11. Specify the signal names for the Verilog net type `Supply 1`.

12. Click *OK* to save the settings and to close the *Verilog Netlist* dialog box.

13. Click *OK* to close the *Concept Options* dialog box.

To specify the options for generating the VHDL netlist for synthesizing a design

1. In Concept HDL, choose *Tools > Options*.

The *Concept Options* dialog box appears.

2. Select the *Output* tab.

   Ensure that the *Create Netlist* check box is selected.

3. Select the *VHDL* check box.

4. Click the *Options* button next to the *VHDL* check box.

   The *VHDL Netlist* dialog box appears.

5. Select the *Verbose Output* check box to log the debug messages of the netlisting process in the `hdldir.log` file located in the `<project_directory>/temp/` directory.

6. Select the *Strict Entity Check* check box to enable checking by comparison of instance properties and symbol properties like the `VHDL_MODE` property.

7. Select the *Check InstanceVs Signal* check box if you want Concept HDL to check if the name of any signal on the schematic is the same as `page<page_number>_<value of PATH property on any instance>`. If this check box is selected, Concept HDL displays the following error message for every signal that has the same name as `page<page_number>_<value of PATH property on any instance>`:

   ```
   126 ERROR "Identifier is used as both a PATH value and a signal name."
   ```

8. Specify the maximum number of netlisting errors that you want to allow in the *Max Errors* field. If the number of netlisting errors in the design exceeds the number specified here, Concept HDL will not generate the netlist.

9. Specify the VHDL logic type for all the vectored ports and signals in the design. You can specify any legal VHDL vector type, such as `STD_LOGIC_VECTOR` and `BIT_VECTOR`. The default value is `STD_LOGIC_VECTOR`.

   Note: The specified vector type applies to all drawings in the design. You can override the vector type for individual drawings by using the `VHDL_VECTOR_TYPE` property on a `VHDL_DECS` symbol.

10. Specify the VHDL logic type for all scalar ports and signals in the design. You can specify any legal VHDL scalar type, such as `STD_LOGIC` and `BIT`. The default value is `STD_LOGIC`.

    Note: The specified scalar type applies to all drawings in the design. You can override the scalar type for individual drawings by using the `VHDL_SCALAR_TYPE` property on a `VHDL_DECS` symbol. For more information on specifying the VHDL logic type of ports and signals, see Setting the VHDL Logic Type for Ports and Signals on page 160.

11. Specify the names of the libraries that are to be used in VHDL library clauses in the VHDL entity and architecture text generated from the schematic. If you do not specify a library, `IEEE` will be used as the default library.

Note: The default library (`IEEE`) is not used if you specify any other library. If you want to use the `IEEE` library along with other libraries, you must explicitly add the `IEEE` library.

You can also add libraries for a drawing by using the `LIBRARY` property on a `VHDL_DECS` symbol. In the VHDL entity and architecture text generated from the schematic, the libraries on the symbol will be appended to the list of libraries you specify here.

12. Specify the names that are to be used in VHDL use clauses in the VHDL entity and architecture text generated from the schematic. There is no limit on the number of use clauses you can add. If you do not specify any use clauses, `IEEE.STD_LOGIC_1164.ALL` will be used as the default.

Note: The `IEEE.STD_LOGIC_1164.ALL` use clause will not be used if you add any other use clauses. If you want to use `IEEE.STD_LOGIC_1164.ALL` along with any other use clause, you must explicitly add `IEEE.STD_LOGIC_1164.ALL`.

You can also add use clauses for a drawing by using the `USE` property on a `VHDL_DECS` symbol. In the VHDL entity and architecture text generated from the schematic, the use clauses on the symbol will be appended to the list of use clauses you specify here.

13. Click *OK* to save the settings and close the *VHDL Netlist* dialog box.

14. Select the *Annotate Synthesis Constraints in Netlist* check box if you want the synthesis constraints specified in Concept HDL to be written in the netlist.

Synthesis constraints are specified in Concept HDL using vendor-specific (Xilinx, Altera, or Actel) properties. For more information, see the *Programmable IC Online Help.*

15. Click *OK* to close the *Concept Options* dialog box.

To generate the netlist for synthesizing a design

➤ In Concept HDL, choose *File > Save*.

The Verilog netlist file `verilog.v` and VHDL netlist file `vhdl.vhd` are generated in the following directory:

```
<project_directory>/worklib/<design_name>/sch_1/
```

Important

If you are working on a large design, saving the design in Concept HDL takes a long time because the netlist for the entire design is regenerated every time you save the design. You can avoid this by disabling netlisting of the design. For more information, see Disabling Netlisting of Designs on page 374.

The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box in Concept HDL. Click on an error to highlight the area on the schematic where the error has ocurred. Correct the schematic errors and generate the netlist again.

The schematic errors that are identified during the process of generating the netlist for packaging are stored in the `netlister.mkr` file that is located in the `<project_directory>/temp/xxnedtmp/` directory. You can view the errors by loading the file in the *Markers* dialog box. The `hdldir.log` file located in the `<project_directory>/temp/` directory contains the details of the netlisting process.

To view netlisting errors

1. In Concept HDL, choose *Tools* > *Markers*.

   The *Markers* dialog box appears.

2. Choose *File* > *Load* and open the `netlister.mkr` file located in the `<project_directory>/temp/xxnedtmp/` directory.

   The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box.

Netlisting for Analog and Mixed Signal Simulation

Concept HDL 14.0 supports analog and mixed signal simulation using the PSpice A/D simulator. PSpice A/D is a simulation program that models the behaviour of a design containing any mix of analog and digital devices.

To generate the netlist for PSpice simulation

➤ In Concept HDL, choose *PSpice* > *Create Netlist*.

A netlist file <design_name-design_name>.net will be generated in the following directory:

`<project_directory>\worklib\<design_name>\cfg_analog\`

For more information, see the *PSpice A/D User Guide* and *PSpice A/D Reference*.

The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box in Concept HDL. Click on an error to highlight the area on the schematic where the error has ocurred. Correct the schematic errors and generate the netlist again.

The schematic errors that are identified during the process of generating the netlist for packaging are stored in the `<design_name>.mkr` file that is located in the `<project_directory>\worklib\<design_name>\cfg_analog\` directory. You can view the errors by loading the file in the *Markers* dialog box. The `hdldir.log` file located in the `<project_directory>/temp/` directory contains the details of the netlisting process.

To view netlisting errors in the PSpice netlist

1. In Concept HDL, choose *Tools* > *Markers*.

   The *Markers* dialog box appears.

2. Choose *File* > *Load* and open the `<design_name>.mkr` file located in the `<project_directory>\worklib\<design_name>\cfg_analog\` directory.

   The schematic errors that are identified during the netlisting process are displayed in the *Markers* dialog box.

Netlisting Read-only Blocks Used in a Design

When the design is netlisted, Concept HDL generates files in the libraries used in the design. If you use blocks from read-only libraries in your design, Concept HDL may try to write files in the read-only libraries when the design is netlisted . If this occurs, the following error message is displayed:

```
NTL_ERROR : Need to analyze the design for cell : <cell name> in library :
<library name>", root->cellName, root->libName

Cell : <cell name> in library : <library name> is marked as read only", root-
>cellName, root->libName

Please provide write permissions in the cell for analysing the design.

You can use the TMP directive in cds.lib file. Please create the cell in the
TMP library.
```

Do one of the following:

- Provide write permissions in the cell containing the read-only block
- Use the TMP attribute in the `cds.lib` file to specify a temp location for the library containing the read-only block. The TMP attribute is used in the `cds.lib` file as below:

  ```
  ASSIGN <name of library containing read-only block> TMP <name of temp
  directory>
  ```

The usage of the TMP attribute is explained using the following figure.

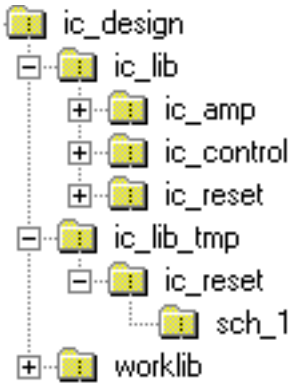
Directory Structure of a Project

The library `ic_lib` is a read-only library. The cell `ic_reset` contains the read-only block used in the design `ic_design`. When the design is netlisted, if Concept HDL displays the error message listed above, use the TMP attribute in the `cds.lib` file as below:

```
DEFINE ic_lib ./ic_lib
ASSIGN ic_lib TMP ./ic_lib_tmp
```

The first entry defines the `ic_lib` library. The second entry assigns the attribute TMP to the library defined as `ic_lib`. The value of TMP is `./ic_lib_tmp`.

The following figure displays the directory structure after the design is re-netlisted.



Note that Concept HDL has created the following:

- A directory `ic_lib_temp`

  This is the temp location for the read-only library `ic_lib`.

- a directory `ic_reset` under `ic_lib_temp`

  This is the temp location for the cell `ic_reset` that contains the read-only block used in the design.

- a directory `sch_1` under `ic_reset`

  This is the sch_1 view for the cell `ic_reset`. All the files generated by Concept HDL when the read-only block `ic_reset` is netlisted are written to the specific views.

16

Plotting Your Design

The plotting facility enables you to make hardcopies of your designs for debugging or documentation. You can use any of the plotters that are configured with your system. You can take plots on various global or local paper sizes. Select different orientations, scalings and other options. You can also customize plotting at the project level or at the site level.

Depending upon the methods used to plot and customize designs, Concept HDL provides the following plotting modes:

- Windows Plotting on Windows and UNIX Platforms
- HPF Plotting on UNIX Platforms
- Hierarchical Plotting

Windows Plotting on Windows and UNIX Platforms

Although this facility is available on both NT and UNIX platforms, it is referred to as Windows plotting because it uses Windows services to generate the plot output. Also, it uses the Windows style of storing information about plotters in the `win.ini` file or the registry. You can customize the plotter settings according to your needs by making some changes in the `win.ini` file or registry.

Project and site level customization of some aspects of plotting can be done through project directives also.

For details on customization of Windows plotting, refer to Customization of Plotting on UNIX Platforms on page 433.

Note: If you want to plot cross-referencing information on your design, first run the *Cross Referencer* tool on your design. The tool generates signal and part cross references for flat and hierarchical schematic drawings. It places the signal cross references directly on the page where the signals appear and creates text reports that contain the list of signal and part cross references. You can then plot the design to view all the additional information that has been attached to the design.

For plotting a design, you may follow the following steps:

1. Setting Up Windows Plotting Options
2. Previewing the Design
3. Plotting the Design

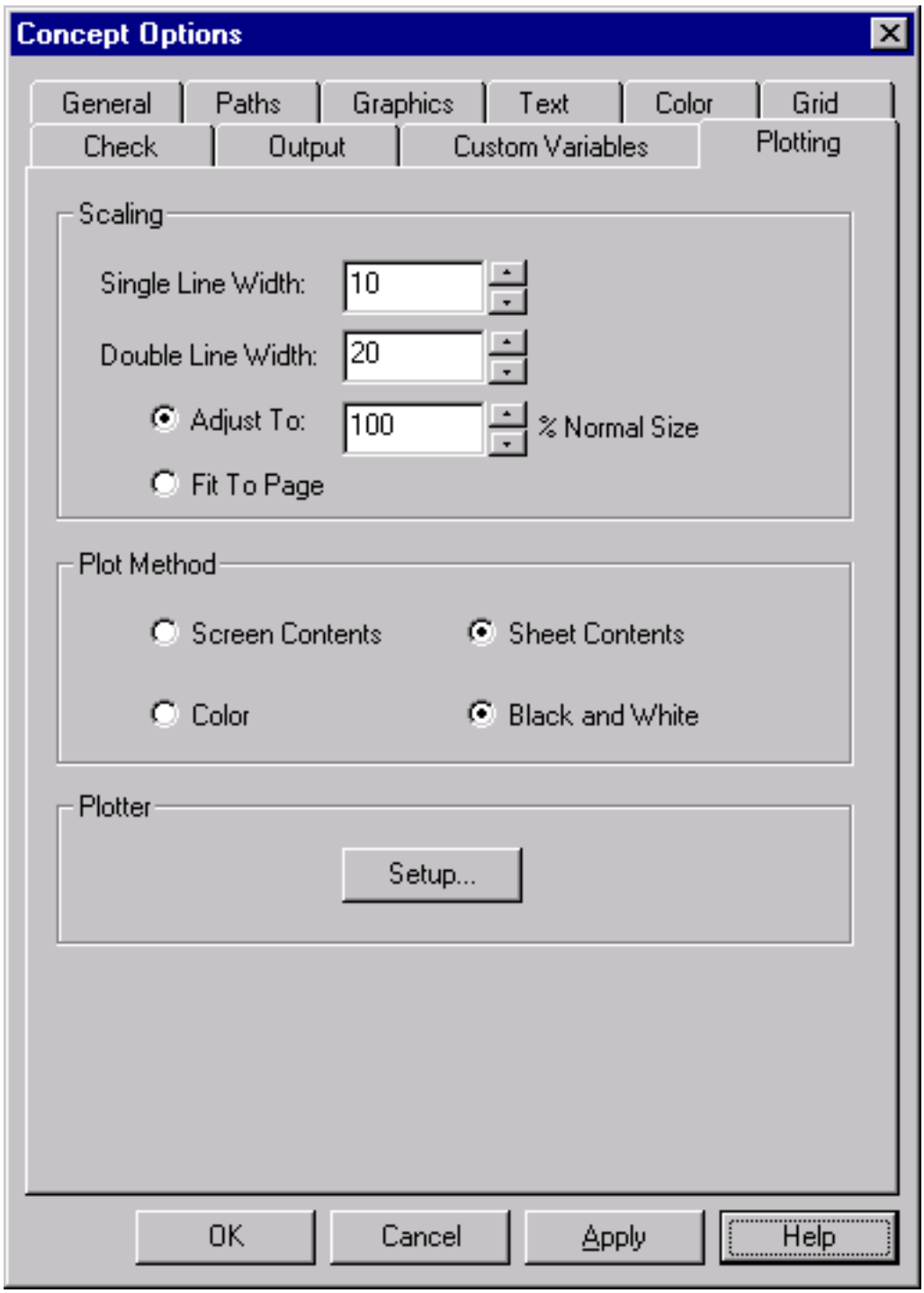
Setting Up Windows Plotting Options

Before setting up the Windows plotting options, ensure that the plotter you want to use is configured properly.

You can setup the Windows plotting options in the *Plotting* tab of the *Concept Options* dialog box.

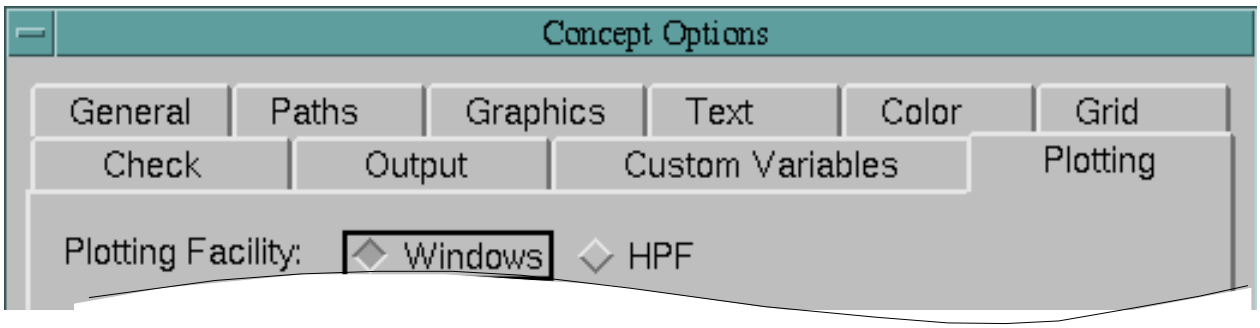
1. Choose *File > Plot Setup* or *Tools > Options > Plotting*.

The *Concept Options* dialog box appears.



Note: On UNIX, additional options are given, to choose between Windows and HPF

plotting. Select *Windows*.



2. Specify the width of single lines in the *Single Line Width* field.

   This specifies the width of lines used to draw thin wires, boundaries of components and text on schematics. By default, the single line width is 1.

3. Specify the width of double lines in the *Double Line Width* field.

   This specifies the width of lines used to draw buses and thick wires on schematics. By default, the double line width is 10.

4. To adjust the plot size, choose *Adjust To* or *Fit To Page*.

   - If you select *Adjust To*, specify the percentage by which to increase or decrease the size of the drawing. Concept HDL then plots the drawing on one or more papers of the specified size. The paper size can be specified by clicking the *Setup* button.

     For example, if you have a drawing with Cadence A size page border, the percentage specified is 100, and the paper size selected is A4. The Cadence A size page border is bigger in size than A4. So, the schematic is plotted on more than one A4 paper.

   - If you select *Fit To Page*, Concept HDL adjusts the size of the drawing so that it fits into one page of the specified paper size.

     For example, you may have a drawing with Cadence A size page border, and the paper size selected is A4. Even though the Cadence A size page border is bigger in size than A4, the schematic is plotted so that it fits on one A4 paper.

5. To select the plot method, choose

   a. *Screen Contents* or *Sheet Contents*.

   If you choose *Screen Contents*, Concept HDL plots the portion of the schematic that is displayed on the screen.

   If you choose *Sheet Contents*, Concept HDL plots the entire page.

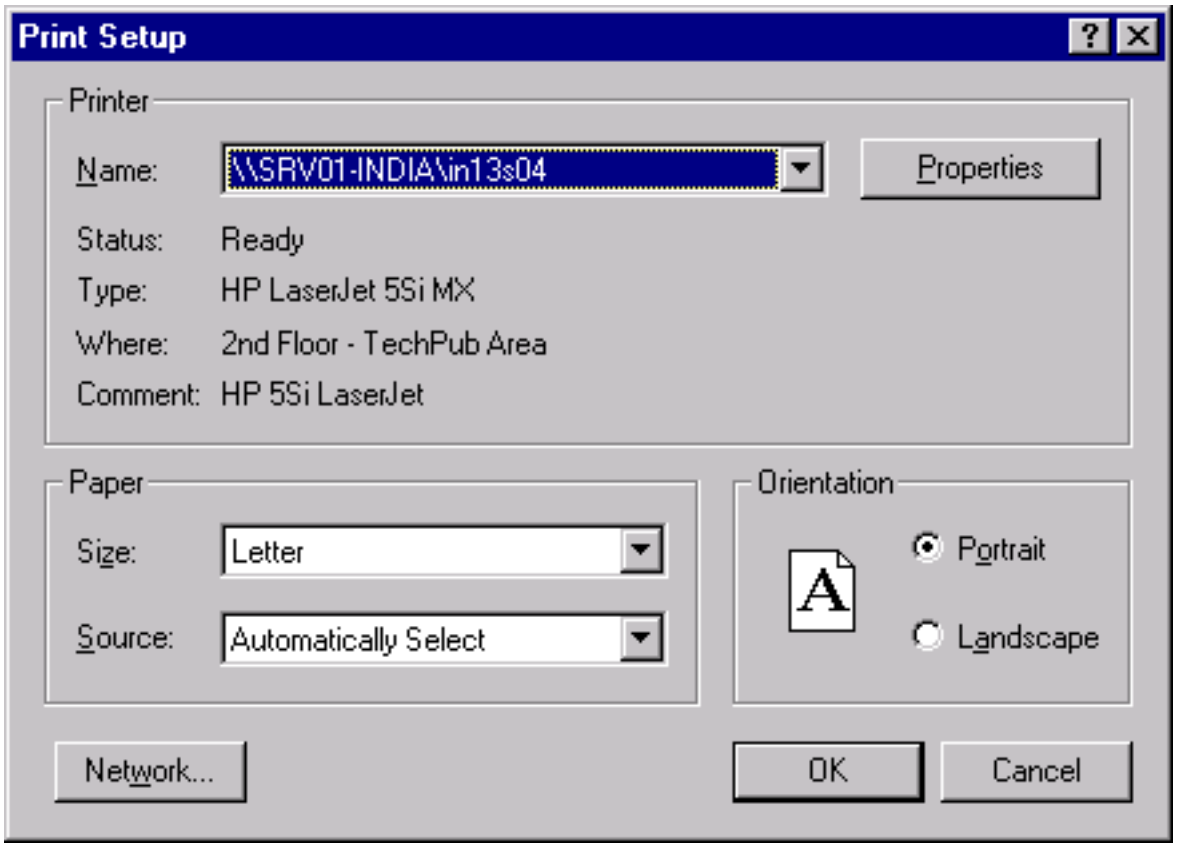
b. *Color* or *Black and White.*

If you choose *Color* and are using a color plotter, Concept HDL plots the drawing in color. It plots in gray scales if you are using a black and white printer.

If you choose *Black and White*, Concept HDL plots the drawing in black and white.

6. To set up the plotter, click *Setup*.

The *Print Setup* dialog box appears.



7. Choose the name of the plotter from the drop-down list.

The list shows the plotters that are configured with the system.

8. Click *Properties*.

A dialog box appears showing the system's default settings for plotting. Do not change any settings here. On UNIX, this button should be used to specify a filename in case you want to plot to a file.

9. Choose the paper size.

The default paper size is same as that of the system. You should use only those paper sizes that are supported by the plotter you have chosen to plot the design.

10. Choose the source of paper.

The default option is the same as that of the system.

11. To plot to a shared printer, click *Network*.

The *Connect to Printer* dialog box appears. It displays a list of network plotters from which you can select a plotter. The plot output is directed to the selected plotter.

12. Choose *Portrait* or *Landscape* as the orientation of the plot output.

13. Click *OK*.

The *Print Setup* dialog box closes.

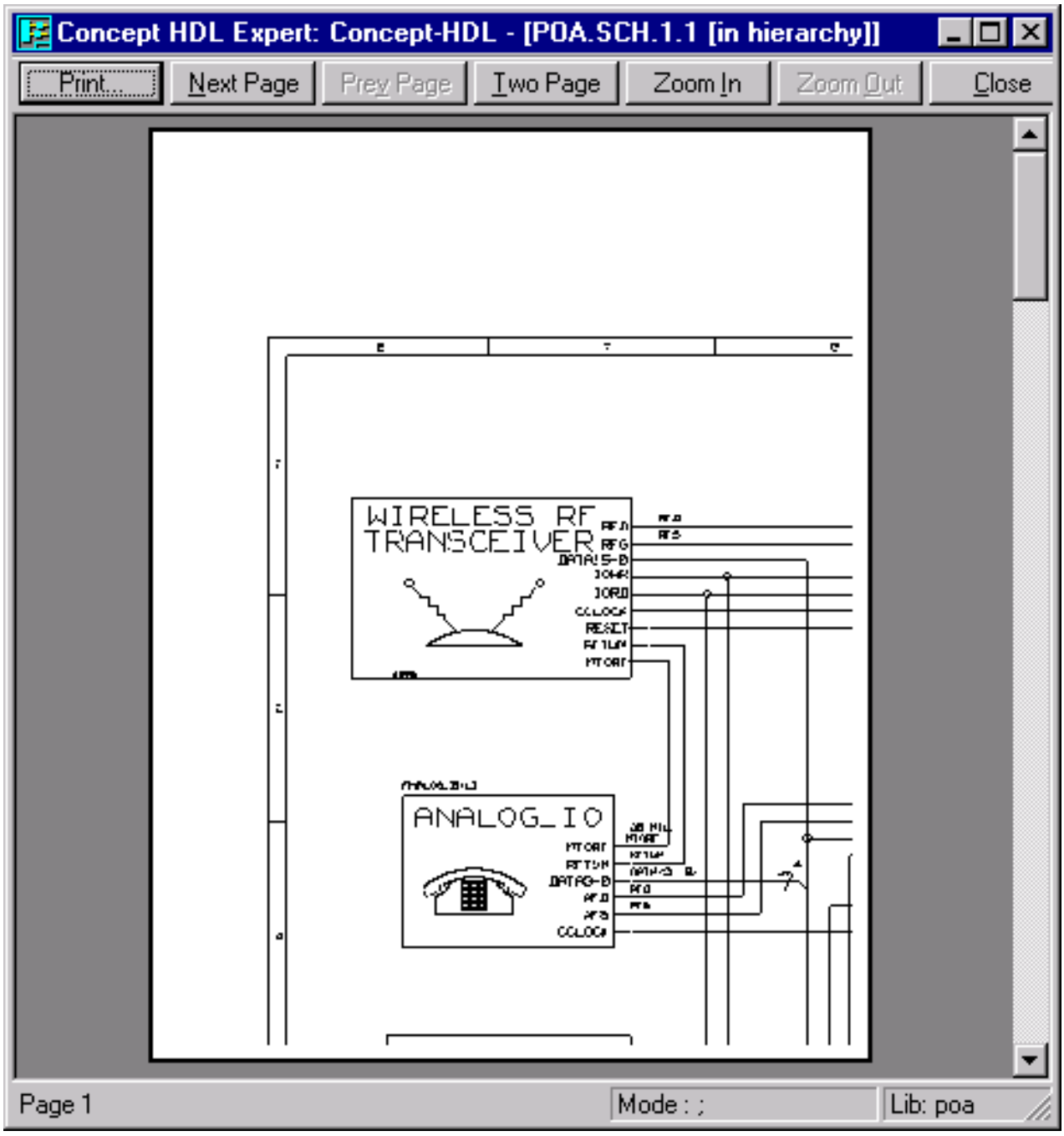
14. Click *OK*.

The *Concept Options* dialog box closes,* and all the settings are saved in the project (.cpm) file.

Previewing the Design

1. Choose *File > Plot Preview*.

The Preview window appears.



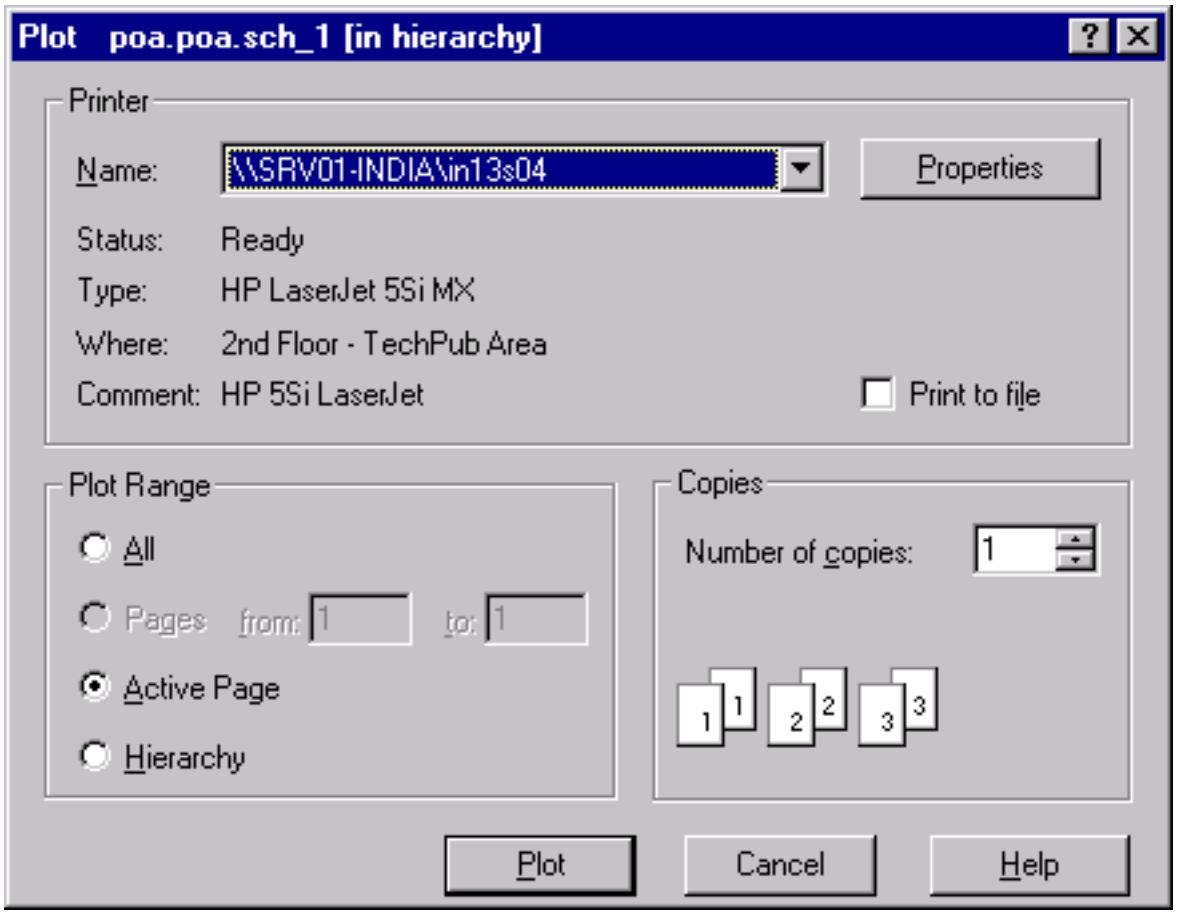
You can click the *Two Page* button to display two pages side by side, zoom in and zoom out the design.

Plotting the Design

1. Choose *File > Plot.*

   The *Plot* dialog box appears.

   .



   Note: The *Hierarchy* button does not appear if you are out of the hierarchy.

2. Choose the plotter name if you do not want to use the default plotter.

3. Check *Print to File* to plot the drawing to a file. The name of the file can be given in the Print to File dialog box which appears when you click Plot if the *Print to File* button is checked.

   Note: On UNIX, this check box does not appear. To plot to a file on UNIX, click the *Properties* button. Select the *File* or *Encapsulated PostScript File* option and specify the name of the file.

4. To select the plot range, choose *All*, *Pages from*, *Active Page*, or *Hierarchy*.

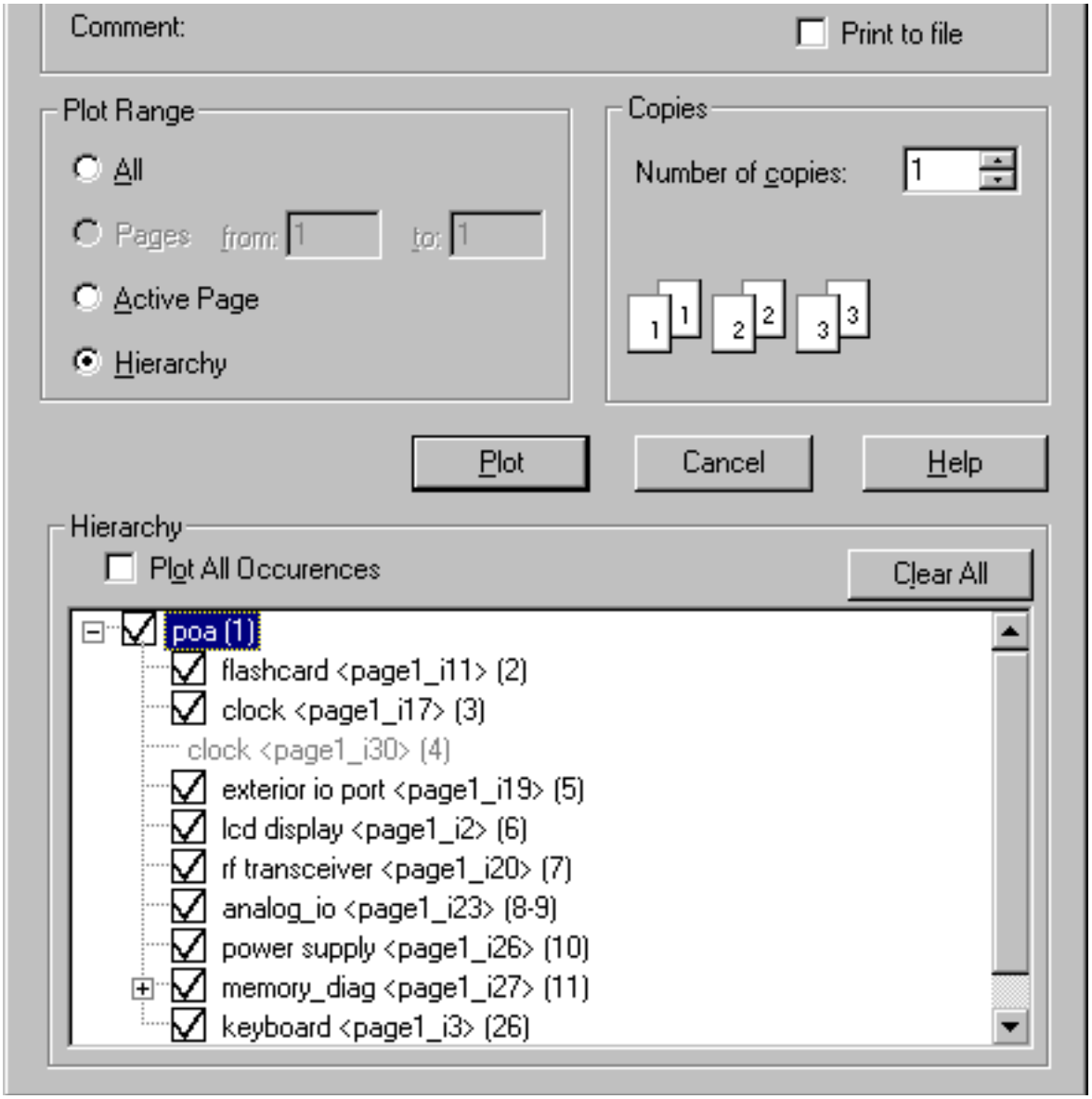
- If you choose *All*, Concept HDL plots all the pages in your currently opened design in the active viewport.

Note: To plot all pages of the design, Concept HDL counts the number of .csb files. This gives the number of pages in the design.

- If you choose *Pages from*, specify the range of pages for plotting.
- If you choose *Active Page*, Concept HDL plots the current page of the design opened in the active viewport.

- If you choose *Hierarchy*, Concept HDL extends the Plot dialog box to display the hierarchical structure of the entire design. You can select or deselect sub-designs for plotting. You can click *Clear All* to clear all selections.

.



For more information on hierarchical plotting, see Hierarchical Plotting on page 424.

5. Click *Plot*.

Concept HDL plots the drawing.

Plotting in Batch Mode

The plot settings can be set up in batch mode for:

- The current session
- All sessions

Important

For setting options in the current session, see Setup Commands and for setting options for all sessions, see Project File Directives.

Console Commands on NT and UNIX

Setup Commands

Command	Capability
SET WPLOT_SPOOLed	Sets the plot output to be directed to a file.
SET WPLOT_FILE [<path>/] <*file_name*>	Sets the filename or path for the plot file you generate. If you do not set a filename, Concept HDL plots to a plot file named `output.ps`. The `set wplot_spooled` command sets the print to file option as the default. To set this option for a given project in all sessions and to specify the name of the generated postscript file, add the following directives manually in the project file (<*project_name*>.cpm): PLOT_TO_FILE ‘YES’ PLOT_FILE_NAME ‘output.ps’ The format of the plot file depends on the plotter driver. The format can be PCL, PS, or any other.
SET WPLOT_LOCAL	Sets the plot output to be directed to a plotter.
SET WPLOT_THIN_width <*width*>	Sets the width of thin lines in plots. The default is 5.
SET WPLOT_THICK_width <*width*>	Sets the width of thick lines in plots. The default is 10.

Command	Capability
SET WPLOT_SCREEN	Sets the plot method to plot screen contents, clipping the drawing outside the display screen.
SET WPLOT_SHEET	Sets the plot method to plot sheet contents.
SET WPLOT_ADJust	Sets the adjust to scale option.

SET WPLOT_SCALE <*scale_value*>	Adjusts the scale value of the plot to a given value.
	Note: This command must be preceded by set wplot_adjust.
	Example: If you have set the scale value to 50% in your project file, you can change the scale value to 100% by using the command:
	`set wplot_adjust`
	`set wplot_scale 100`

SET WPLOT_FIT_to_page	Sets the plot to fit the paper size specified.
SET WPLOT_LANDscape	Sets the plot orientation to landscape
SET WPLOT_PORTrait	Sets the plot orientation to portrait
SET WPLOT_DEFault	Instructs Concept HDL to read and set defaults from the directives in the project file.
SET WPLOT_PAPER	Sets the paper size for plotting. To see the standard paper sizes, refer Paper Sizes Supported by Concept HDL on page 401. Example: `set wplot_paper A4` If you set a paper size not supported by the current printer, an error message is displayed.
SET WPLOT_PLOTTER <*plotter_name*>	Sets the plotter to the name specified. If the plotter name consists of any special characters, precede the first such character with a \ and put the plotter name within quotes.

Project File Directives

These directives correspond to some fields in the *Plot Setup* dialog box. The directives are read and written by the *Plot Setup* dialog box. You can also change the values of these directives in the project file (.cpm) without invoking the *Plot Setup* dialog box.

Directive	Default Value	Description
PLOT_SINGLE_WIDTH	1	This directive corresponds to the single line width field in the *Plot* dialog box. This specifies the width of thin wires and buses. You can also control the text width by this field. On some plotters, if the plot output is very thin and does not show the text clearly, this width can be increased making the whole design, along with the text, thicker.
PLOT_DOUBLE_WIDTH	10	This directive corresponds to the double line width field in the Plot dialog box. It specifies the width of thick wires and buses.
PLOT_SCALE	100	This directive corresponds to the scale value specified in the Adjust to field in Plot setup. It specifies the percentage by which to increase or decrease the plot size.
PLOT_FIT_TO_PAGE	OFF	This directive, when set to ON, directs Concept HDL to adjust the plot according to page size.
PLOT_SCREEN	OFF	This directive, when set to ON, directs Concept HDL to plot the portion of the schematic that is displayed on the screen.
PLOT_COLOR	OFF	This directive, when set to ON, directs Concept HDL to plot the drawing in color if you are using a color plotter, and in gray scales if you are using a black and white printer.
PAPER_ORIENTATION	2	This directive sets the orientation of the plot output. You can set it to Portrait or Landscape.

Directive	Default Value	Description
PAPER_SIZE	9	The number shows the index (0-based) number of the paper size selected in the combo box in the *Plot Setup* dialog box. This directive is useful if you want to take plot in one paper size only and want to retain this setting over multiple Concept sessions. It is recommended that you set the paper size in the *Plot Setup* dialog box. If you need to change it manually, ensure that you enter the correct index to map to the correct paper size in the list displayed in the combo box. All plotters may not support all the available sizes.
PAPER_SOURCE	4	The number shows the index (0-based) number of the paper source selected in the combo box in the *Plot Setup* dialog box. This directive is useful if you want to use one paper source throughout the site or for all your designs. It is recommended that you change the paper size through the *Plot Setup* dialog box. If you need to change it manually, ensure that you enter the correct index to map to the correct paper source in the list displayed in the combo box. **Note:** All plotters may not support all the available paper sources.

Plot Command

The syntax of the plot command is

```
plot [<lib>].[<cell>].[<view>].[<ver>].[<page>]
```

Examples:

Command	Capability
Plot	Plots the currently opened drawing
Plot cache	Plots all pages in the cache
Plot cache.sym.1.1	Plots the symbol view in the cache

Command	Capability
Plot cache.sym.1.2	Plots the page 2 schematic of the cache
Plot cache.sch.1.*	Plots all pages of version 1

where cache is the name of the drawing.

Note: Wildcard characters are supported for page numbers only.

There are two directives for the plot console command. These are not read or written by plot setup of Concept HDL. You should always change them manually in the `.cpm` file.

Directive	Default Value	Description
PLOT_TO_FILE	NO	This directive tells the plot command to direct its plot output to a file. The file will by default be `output.ps`. Possible values: NO YES
PLOT_FILE_NAME		This directive is used in conjunction with the PLOT_TO_FILE directive to change the default file name for the plot output. You can also specify the full path of the file if you wish to direct output to a directory other than the project directory. By default the file name is `output.ps` and it is created in the project directory.

Paper Sizes Supported by Concept HDL

The possible paper sizes that you can use are
:

LETTER	LETTERSMALL	EXECUTIVE
LEDGER	LEGAL	STATEMENT
A3	A4	A4SMALL
B4	B5	FOLIO
10X14	11X17	NOTE
ENV_10	ENV_11	ENV_12

CSHEET	DSHEET	ESHEET
ENV_C5	ENV_C3	ENV_C4
ENV_C65	ENV_B4	ENV_B6
FANFOLD_STD_GERMAN	ENV_MONARCH	ENV_PERSONAL
FANFOLD_LGL_GERMAN	ENV_B5	A5
FANFOLD_US	ENV_ITALY	ENV_DL
QUARTO	ENV_C6	ENV_14
ENV_9		

Note: You should use only those paper sizes that are supported by the plotter you have chosen to plot the design on.

HPF Plotting on UNIX Platforms

From Concept HDL, you can either plot a drawing directly or create a plot file to be printed at a later time or to be physically transferred to another system. In either case, you use the `hardcopy` command.

Concept HDL calls a utility called `hpfhdl` to perform its plots. You can also run `hpfhdl` separately from a UNIX shell window.

You can plot drawings in the HPF plotting mode

- From Concept HDL using the *HPF Plot* dialog box

  For more information, see Plotting the Design on page 408.

- From Concept HDL using the `hardcopy` console command

  For more information, see Plotting the Design from the Console Window on page 410.

- From a UNIX shell using the `hpfhdl` utility

  You cannot plot occurrence properties or hierarchical drawings using the `hpfhdl` utility.

  For more information, see Plotting Drawings from the UNIX Shell on page 418.

Setting up HPF Plotting Options

Before setting up the HPF plotting options, ensure that the plotter you want to use is configured properly. Also, ensure that the plotter is defined in the plotting configuration file `.cdsplotinit`. For more information on configuring your system for plotting, see the *Plotter Configuration User Guide*.

You can set up the HPF plotting options in the *Plotting* tab of the *Concept Options* dialog box. To access the *Plotting* tab of the *Concept Options* dialog box for setting up HPF plotting options, do one of the following:

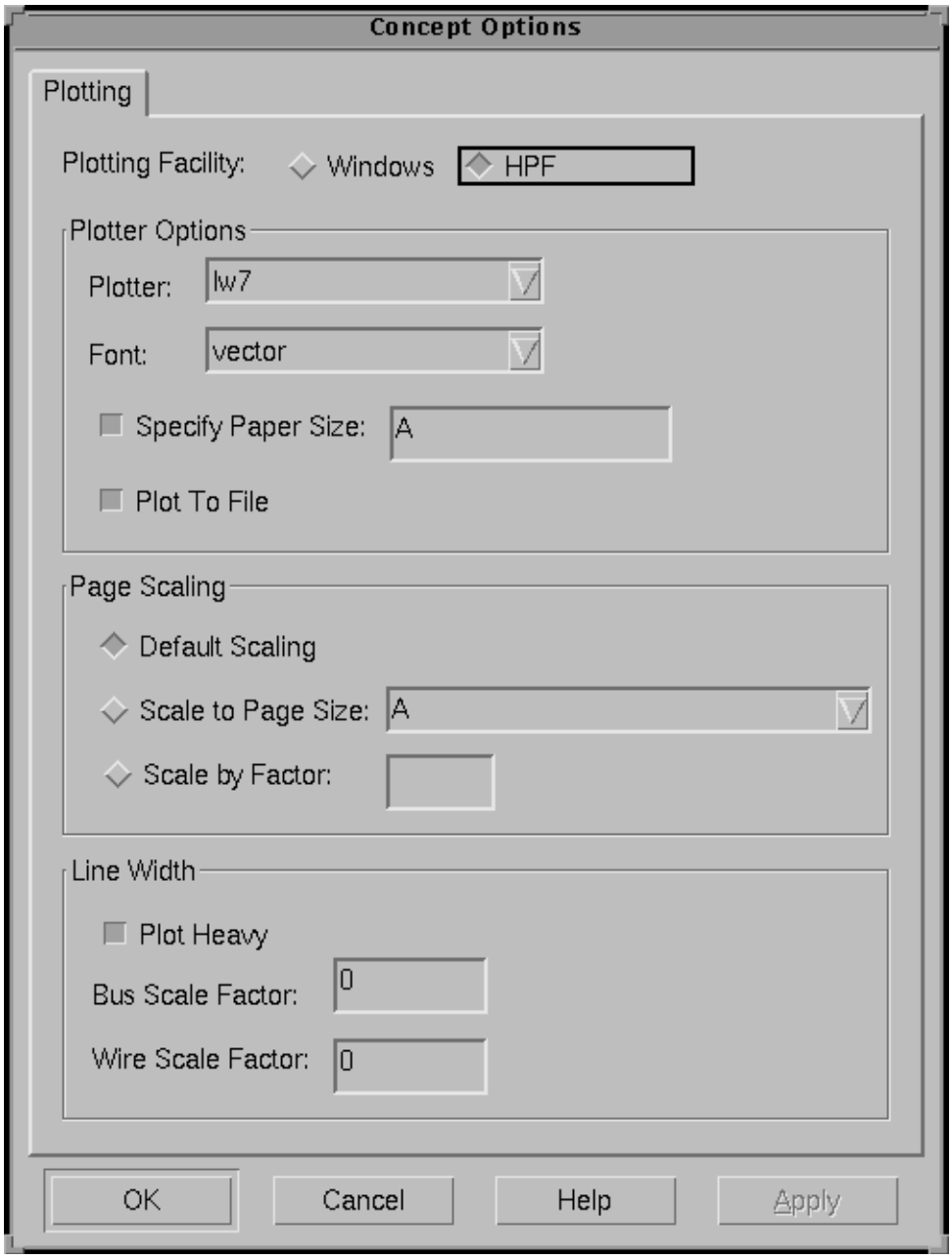
- From the *File* menu,
  1. Choose *File > Plot Setup*.

     The *Plotting* tab of the *Concept Options* dialog box appears.
  2. Select *HPF* as the plotting facility.
- From the *Tools* menu,
  1. Choose *Tools > Options.*

     The *Concept Options* dialog box appears.
  2. Select the *Plotting* tab.
  3. Select *HPF* as the plotting facility.
- From the *HPF Plot* dialog box,
  1. Choose *File > Plot*.

     The *HPF Plot* dialog box appears.
  2. Click *Setup*.

The setup options for HPF plotting appear.



1. Select the plotter you want to use in the *Plotter* drop-down list.

   The *Plotter* drop-down list displays all the plotters that are defined in the plotting configuration file `.cdsplotinit`. For example, if the `.cdsplotinit` file has the following entries for the Hewlett-Packard 7600 Series Electrostatic plotter, `hpgl2` will be displayed in the *Plotter* drop-down list.

   ```
   hpgl2|Hewlett-Packard 7600 Series Electrostatic: \
   ```

```
:manufacturer=Hewlett-Packard: \
:type=hpgl2: \
:maximumPages#10: \
:resolution#1016: \
:paperSize="A" 9816 8236:    \
:paperSize="D" 34544 22352: \
```

The first plotter defined in the `.cdsplotinit` file is the default plotter. For more information on the `.cdsplotinit` file, see Customization of HPF Plotting on UNIX Platforms on page 436.

The plotter that you select in the *Plotter* drop-down list is written in the `<project_name>.cpm` file by using the `HPF_PLOTTER '<plotter_name>'` directive. If you delete the entries for the plotter from the `.cdsplotinit` file, you will not be able to plot the drawing unless you select another plotter from the *Plotter* drop-down list.

2. Select the font to be used for plotting in the *Font* drop-down list.

   The text in the drawing is plotted using the selected font. The default font is `VECTOR`. The following fonts are supported:

   - VECTOR
   - CURSIVE
   - NATIVE
   - GOTHIC
   - SYMBOL
   - GREEK
   - VALID
   - MILSPEC

3. Select the check box next to the *Specify Page Size* field and specify the paper size.

   If the paper size name has spaces, enclose it in parentheses. For example, if the paper size name is `22 inches wide`, specify the paper size as "`22 inches wide`".

   The paper size that you specify must be defined for the plotter in the `.cdsplotinit` file. For example, if the entries for the `hpgl2` plotter in the `.cdsplotinit` file are as below, you can specify `A`, `D`, `E`, "`22 inches wide`" or "`34 inches wide`" as the paper size.

```
hpgl2|Hewlett-Packard 7600 Series Electrostatic: \
     :manufacturer=Hewlett-Packard: \
```

```
:type=hpgl2: \
:maximumPages#10: \
:resolution#1016: \
:paperSize="A" 9816 8236:    \
:paperSize="D" 34544 22352: \
:paperSize="E" 44704 34544: \
:paperSize="22 inches wide" 0 22352: \
:paperSize="34 inches wide" 0 34544:
```

You can define the paper size name for a plotter by using the `paperSize` option in the `.cdsplotinit` file.

Note: If you do not select the check box next to the *Specify Page Size* field, the first paper size specified in the `.cdsplotinit` file for the specified plotter is taken as the default paper size. In the above example for the `hpgl2` plotter, `A` will be taken as the default paper size if you do not select the check box next to the *Specify Page Size* field.

4. Select the *Plot to File* check box if you want to print the drawing to a `vw.spool` file. The `vw.spool` file will be created in the project directory when you plot the drawing.

   Note: If this check box is not selected, the drawing will be plotted.

5. To scale the drawing, select *Default Scaling*, *Scale to Page Size*, or *Scale by Factor*.

Select	To
Default Scaling	Plot the drawing as it is. In other words, the default scale factor `1` is used.

Select	To
Scale to Page Size	Scale the drawing to be plotted to the page size you select. **Note:** The paper size that you select need not be defined for the plotter in the `.cdsplotinit` file. This option is similar to the *Fit To Page* option in the Windows plotting facility with the added feature of specifying the page size to which the drawing has to fit. For example, if you have used the `C SIZE PAGE` page border (17 x 22 inch) symbol in your drawing and want to plot the drawing in `A` paper size (8 1/2 x 11 inch), specify `A` in the *Specify Page Size* field and select `A` in the *Scale to Page Size* drop-down list. **Note:** If the paper size you select in the *Scale to Page Size* field is not the same or smaller than the paper size you specified in the *Specify Page Size* field, the drawing will be plotted in multiple sheets. For example, if you have used the `C SIZE PAGE` page border symbol in your drawing and plot the drawing to paper size `B`, the drawing will be plotted in multiple sheets of paper size `B` if you do not select paper size `B` or a smaller paper size in the *Scale to Page Size* drop-down list.
Scale by Factor	Specify the factor by which you want to scale the plot. For example, a scale factor of 0.5 will create a plot size that is half the drawing size. **Note:** If the scale factor results in a plot size that is larger than the paper size you have specified in the *Specify Page Size* field, the drawing will be plotted in multiple sheets of the paper size you specified in the *Specify Page Size* field.

6. Select *Plot Heavy* if you want to increase line widths of buses and wires.
7. Specify the scale factor in the *Bus Scale Factor* field to increase or decrease the line widths of thick wires (vectored signals) in plots.
8. Specify the scale factor in the *Wire Scale Factor* field to increase or decrease the line width of thin wires (scalar signals) and thickness of text in plots.
9. Click *OK*.

Plotting the Design

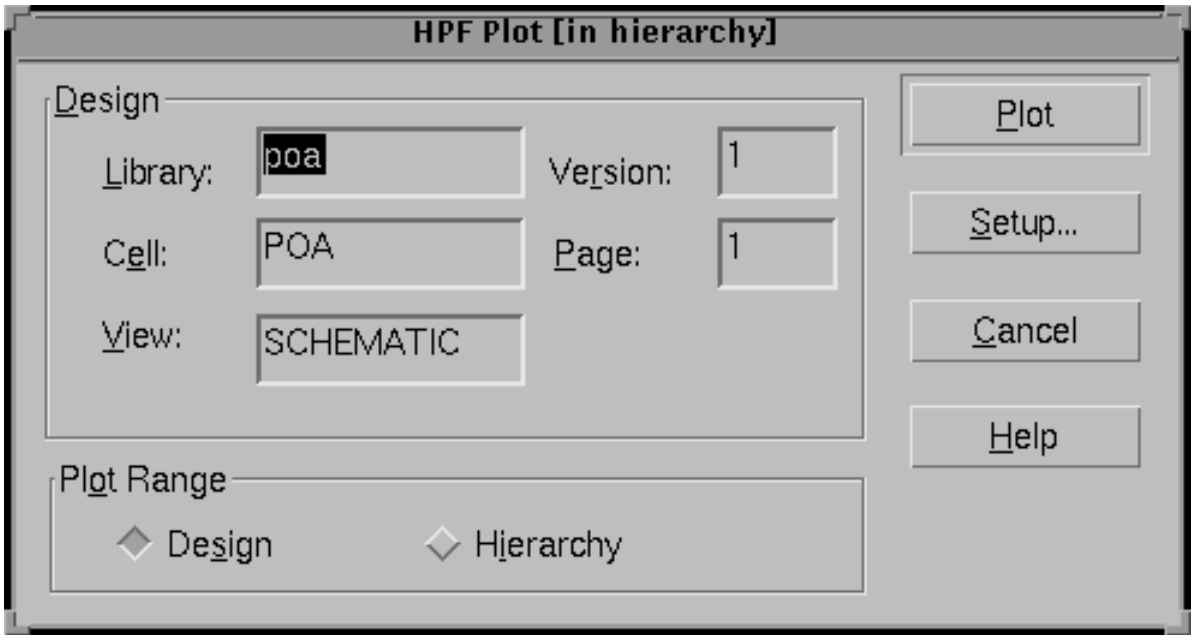
To plot the design, you have to open the *HPF Plot* dialog box. The *HPF Plot* dialog box allows you to plot specific drawings in the design or to select sub-designs from a hierarchical design for plotting.

Note: If you want to plot occurrence properties, you must change to the Occurrence Edit mode before plotting the schematic. For more information, see Occurrence Edit Mode on page 114.

1. Choose *File > Plot*.

   The *HPF Plot* dialog box appears if you selected *HPF* as the plotting facility in the *Plotting* tab of the *Concept Options* dialog box. For more information, see Setting up HPF Plotting Options on page 403.

   The default values are for the current drawing.



2. Click *Setup* if you want to change the HPF plotting options.

   For more information, see Setting up HPF Plotting Options on page 403.

3. In the *Plot Range* group box.
   - Select *Design* if you want to plot the drawing specified in the *Design* group box.
   - Select *Hierarchy* if you want to perform hierarchical plotting.

Concept HDL extends the *HPF Plot* dialog box to display the hierarchical structure of the root design. You can select or deselect sub-designs for plotting. For more information on hierarchical plotting, see Hierarchical Plotting on page 424.

If you select *Hierarchy*, all the fields in the *Design* group box are disabled.

4. Change the library name in the *Library* field, if required.

   If you change the library name, ensure that the library is defined in the `cds.lib` file.

   This field is disabled if you are in the Occurrence Edit mode. In the Occurrence Edit mode, you can plot only the currently active drawing or drawings from the same version of the view in which the drawing exists.

5. Change the cell name in the *Cell* field, if required.

   If you change the cell name, ensure that the cell is present in the library you have specified in the *Library* field. You can use wildcards (* and ?) in this field.

   This field is disabled if you are in Occurrence Edit mode. In Occurrence Edit mode you can plot only the currently active drawing or drawings from the same version of the view in which the drawing exists.

6. Change the view name in the *View* field, if required.

Specify	If
SCHEMATIC	You want to plot schematic drawings
SYM	You want to plot symbol drawings
SCHCREF_1	You want to plot the schematic drawings generated by CRefer in the `schcref_1` view

   If you change the view name, ensure that the view is present in the cell you specified in the *Cell* field. You can use wildcards (* and ?) in this field.

   This field is disabled if you are in the Occurrence Edit mode. In the Occurrence Edit mode, you can plot only the currently active drawing or drawings from the same version of the view in which the drawing exists.

7. Change the version number in the *Version* field, if required.

   The version number indicates the version of the view you want to plot. For example, if you want to plot the schematic drawings in the `sch_1` view of a cell, specify `SCHEMATIC` in the *View* field and `1` in the *Version* field. If you want to plot the symbol drawings in the `sym_3` view of a cell, specify `SYM` in the *View* field and `3` in the *Version* field. To plot the schematic drawings generated by CRefer in the `schcref_1` view of a cell, specify `SCHCREF_1` in the *View* field and `1` in the *Version* field.

By default, the version number is 1. If you change the version number, ensure that the version of the view is present in the cell you specified in the *Cell* field. You can use wildcards (* and ?) in this field.

This field is disabled if you are in the Occurrence Edit mode. In the Occurrence Edit mode, you can plot only the currently active drawing or drawings from the same version of the view in which the drawing exists.

8. Change the page number in the *Page* field, if required.

   By default, the page number is 1. If you change the page number, ensure that the page is present in the version of the view you specified in the *Version* field. You can use wildcards (* and ?) in this field.

9. Click *Plot*.

Plotting the Design from the Console Window

Concept HDL allows you to plot the design in batch mode from the console window. You can setup the HPF plotting options and then plot the design by using the `hardcopy` console command. For more information, see hardcopy Command on page 414.

Note: If you want to plot occurrence properties, you must change to the Occurrence Edit mode before plotting the schematic. For more information, see Occurrence Edit Mode on page 114.

The options for HPF plotting from the console window can be set up for:

- The current session only

  The options you specify for HPF plotting in the *Plotting* tab of the *Concept Options* dialog box are the default plotting options. You can override the default options only for the current session using the `set` console command. To set up HPF plotting options only for the current session, use Setup Commands.

- All sessions

  The options you specify for HPF plotting in the *Plotting* tab of the *Concept Options* dialog box are the default plotting options. To set up the default HPF plotting options for all sessions, use Project File Directives.

Setup Commands

You can setup the following HPF plotting options using the `set` console command. The options that you set using the `set` console command override the default HPF plotting options (setup in the *Plotting* tab of the *Concept Options* dialog box) for the current session.

- Specifying the Font for Plotting on page 411
- Specifying the Default Paper Size on page 411
- Setting Plotting to a Plotter on page 412
- Setting Plotting to a File on page 412
- Specifying the Plotter on page 413

Specifying the Font for Plotting

Console command syntax

```
set font <hpf_font_name>
```

where `hpf_font_name` can be one of the following:

- VECTOR_FONT
- CURSIVE_FONT
- NATIVE_FONT
- GOTHIC_FONT
- SYMBOL_FONT
- GREEK_FONT
- VALID_FONT
- MILSPEC_FONT

Specifying the Default Paper Size

Console command syntax

```
set papersize <option>
```

where `option` is the name (string) indicating the paper size the plotter uses, including any offset.

Example:

```
set papersize A
```

If the paper size name has spaces, enclose it in parentheses. For example, if the paper size name is `22 inches wide`, specify the paper size as "`22 inches wide`".

The paper size that you specify must be defined in the `.cdsplotinit` file. For example, if the entries for the `hpgl2` plotter in the `.cdsplotinit` file are as below, you can specify `A`, `D`, `E`, "`22 inches wide`" or "`34 inches wide`" as the paper size.

```
hpgl2|Hewlett-Packard 7600 Series Electrostatic: \
     :manufacturer=Hewlett-Packard: \
     :type=hpgl2: \
     :maximumPages#10: \
     :resolution#1016: \
     :paperSize="A" 9816 8236:    \
     :paperSize="D" 34544 22352: \
     :paperSize="E" 44704 34544: \
     :paperSize="22 inches wide" 0 22352: \
     :paperSize="34 inches wide" 0 34544:
```

You can define the paper size name for a plotter using the `paperSize` option in the `.cdsplotinit` file.

Note: The first paper size specified in the `.cdsplotinit` file for the specified plotter is taken as the default paper size. In the above example for the `hpgl2` plotter, `A` will be taken as the default paper size.

Setting Plotting to a Plotter

Console command syntax

```
set local
```

or

```
set LOCAl_plot
```

The drawing will be plotted in a plotter.

Setting Plotting to a File

Console command syntax

```
set spooled
```

OR

```
set SPOoled_plot
```

The drawing will be plotted to a file called vw.spool, instead of being plotted on a plotter. The vw.spool file is created in the project directory. This file can be plotted later.

Specifying the Plotter

Console command syntax

```
set PLotter <plotter_name>
```

Where plotter_name is any plotter specified in the .cdsplotinit file.

Project File Directives

The following directives are set in the project file (<projectname>.cpm) when you specify the options for HPF plotting in the *Plotting* tab of the *Concept Options* dialog box.

Directive	Values	Description
HPF_BATCH	Yes No	This directive corresponds to the *Plot to File* check box. It is used to direct Concept HDL to spool the plot to the vw.spool file.
HPF_PLOTTER	Name of the plotter	This directive corresponds to the *Plotter* field. It is used to specify the name of the plotter in the .cdsplotinit file.
HPF_SPEC_PLOT_PAGESIZE	Yes No	This directive corresponds to the check box next to the *Specify Page Size* field. It is used to set the paper size.
HPF_PLOT_PAGESIZE	Page size	This directive corresponds to the *Specify Page Size* field. It specifies the paper on which to plot.
HPF_SCALETYPE	Default, Scale by factor, Scale to Page Size	This directive corresponds to the *Default, Scale by Factor*, and *Scale to Page Size* fields. It specifies the mode of scaling to be used for plotting.

Directive	Values	Description
HPF_PAGESIZE	A B C D E F	This directive corresponds to the *Scale to Page Size* field. It specifies the standard page to which the design gets scaled.
HPF_BUS_SCALEFACTOR	A number	This directive corresponds to the *Bus Scale Factor* field. It specifies the width of buses in the design.
HPF_WIRE_SCALEFACTOR	A number	This directive corresponds to the *Wire Scale Factor* field. It specifies the width of wires, text, and component boundaries in the design.
HPF_FONT	A font name	This directive corresponds to the *Font* field. It specifies the font style to be used for plotting.
HPF_SCALEFACTOR	A number	This directive corresponds to the *Scale by Factor* field. It specifies the scaling factor.

hardcopy Command

You can plot designs in batch mode using the `hardcopy` console command.

Before using the `hardcopy` command, ensure that the plotter you want to use is configured properly. Also, ensure that the plotter is defined in the plotting configuration file `.cdsplotinit`. For more information on configuring your system for plotting, see the *Plotter Configuration User Guide*.

Using the hardcopy Command

The `hardcopy` command takes two arguments, `scale_factor` or `paper_size` and `drawing_name`, in the following syntax:

```
hardcopy[scale_factor|scale_to_page][drawing_name]
```

scale_factor A factor applied to the drawing to determine the final plot size. The default factor is 1.

Specify this option if you want to create a bigger or smaller plot size for the drawing.

- To create a plot size of half the drawing size, use a factor of `.5`.
- To plot twice the drawing size, use a factor of 2.

scale_to_page A pre-determined plot size. The drawing to be plotted is scaled to the page size you specify.

For example, if you have used the `C SIZE PAGE` page border (17 x 22 inch) symbol in your drawing and want to plot the drawing to fit in A paper size (8 1/2 x 11 inch), specify `A` as the value for *scale_to_page*.

The paper size that you specify as the value for this option must be defined for the plotter in the `.cdsplotinit` file. The `paperSize` entry is used to specify the paper sizes that a plotter uses.

If you want the drawing to be scaled to the paper size you specify as the value of the *scale_to_page* option but want to plot the drawing on some other paper size, specify the default paper size using the `set` console command. For more information, see Specifying the Default Paper Size on page 411.

For example, assume that you have entries `A` and `D` for the `paperSize` option for the `hpgl2` plotter in the `.cdsplotinit` file, and you used the `set` console command to specify `D` as the default paper size. If you enter `hardcopy a`, the drawing will be scaled according to the `paperSize` entry `A`, but printed on the `hpgl2` plotter using the paper size specified by the `D` entry.

If you do not specify a default paper size using the `set` console command, the first `paperSize` entry specified for the plotter in the `.cdsplotinit` file is taken as the default paper size.

drawing_name

This is the drawing you want to plot. The drawing does not have to be the one you are currently editing. If you do not specify a drawing name, the current drawing is plotted.

The drawing name is specified using the following syntax:

```
[<library>][<cell>.<view>.<version>.<page>]
```

- where `library` is the name of the library in which the drawing exists.

  You need not specify the library if the `cell` is present in the library containing the root design for the project (also known as `worklib`). The library containing the root design for the project is specified in the *Global* tab of the *Project Setup* window. For more information, see Setting Up a Project on page 74.

- where `view` can be one of the following:
  - `sch` for the schematic view that contains schematic drawings
  - `sym` for the symbol view that contains symbol drawings
  - `schcref_1` for the `schcref_1` view that contains schematic drawings generated by CRefer
- where `version` is the version of the view you want to plot. For example, if you want to plot the schematic drawings in the `sch_1` view of a cell, specify the version as `1`.
- where `page` is the number of the page you want to plot.

You can use wildcards to specify the drawing name. For a better understanding of the syntax used for specifying the drawing name, see Sample hardcopy Commands.

If you are in the Occurrence Edit mode, you can plot only the currently active drawing or drawings from the same version of the view in which the drawing exists. If you specify a different library, cell, view, or version number of the view, Concept HDL will display the following error message when you plot the drawing:

```
Only currently active drawing can be plotted in occurrence
edit mode
```

Sample hardcopy Commands

The following examples assume that you have used the `set` console command to specify `B` as the default paper size. For more information, see Specifying the Default Paper Size on page 411.

Command	Capability
ha	Plots the current drawing at the default scale factor `1` on `B` size paper.
ha b	Scales the current drawing to paper size `B` and plots it on `B` size paper.
ha 1 *	Plots all the schematic drawings, symbol drawings, and the drawings generated by CRefer (in the `CREFOUT` and `SCHCREF_1` views) that are present in the design. The drawings are plotted on `B` size paper.
ha a cache.sch.1.1	Scales the first schematic drawing present in the `sch_1` view of the `cache` cell to paper size `A` and plots the drawing on `B` size paper.
ha 1 <memory>cache.sch.1.1	Plots the first schematic drawing present in the `sch_1` view of the `cache` cell in the `memory` library on `B` size paper.
ha 1 cache.schcref_1.1.1	Plots the first schematic drawing present in the `schcref_1` view of the `cache` cell on `B` size paper.
ha 1 <poa>cache.sch.2.3	Plots the third schematic drawing present in the `sch_2` view of the `cache` cell in the `poa` library. The drawing is plotted on `B` size paper.
ha a <poa>cache.sch.2.*	Scales all the schematic drawings present in the `sch_2` view of the `cache` cell in the `poa` library to paper size `A` and plots the drawings on `B` size paper.
ha 1 cache.sch.*	Plots all the schematic drawings present in all the schematic views (`sch_1`, `sch_2`, `sch_3` and so on) of the `cache` cell. The drawings are plotted on `B` size paper.
ha 1 *.sch.*	Plots all the schematic drawings that are present in all the schematic views (`sch_1`, `sch_2`, `sch_3` etc.) of all cells in all *local* libraries used the project. The drawings are plotted on `B` size paper.

Command	Capability
ha 1 *.schcref_1.*	Plots schematic drawings generated by CRefer in the SCHREF_1 view. The drawings are plotted on B size paper.
ha 2 <lsttl>ls154.sym.1.1	Plots the symbol view of the ls154 part in the lsttl library on B size paper using a scale factor of 2.
ha 2 <lsttl>ls154.*	Plots all versions of the ls154 symbol (the symbol drawings present in the sym_1 and sym_2 views of ls154). The drawings are plotted on B size paper using a scale factor of 2.

Plotting Drawings from the UNIX Shell

You can plot drawings from outside Concept HDL (from a shell window) using the hpfhdl utility. The hpfhdl utility uses the same input file format as the hardcopy command uses within Concept HDL. So, you can plot

- ASCII vectorized format files
- ASCII component (body) files
- Concept HDL binary format files

Important

The hfphdl utility does not allow you to plot custom text and occurrence properties. For more information on custom text and occurrence properties, see Chapter 10, "Working with Properties and Text."

See the following sections for more information:

- Setting Up the hpfhdl Utility on page 419
- Using the hpfhdl Utility on page 419
- Header File Format on page 420
- hpfhdl Command Syntax on page 423

Setting Up the hpfhdl Utility

The system location of the hpfhdl utility is <your_install_dir>/tools/editor/lib/. To run the hpfhdl utility from a UNIX terminal, you have to set environment variables as given below:

On Sun Solaris

```
set path (<your_install_dir>/tools/editor/lib $path)
setenv LD_LIBRARY_PATH <your_install_dir>/tools/lib $LD_LIBRARY_PATH
```

On IBM AIX

```
set path (<your_install_dir>/tools/editor/lib $path)
setenv LIBPATH <your_install_dir>/tools/lib $LIBPATH
```

On HP-UX

```
set path (<your_install_dir>/tools/editor/lib $path)
setenv SHLIB_PATH <your_install_dir>/tools/lib $SHLIB_PATH
```

Using the hpfhdl Utility

You can create a file that you can plot later, or send directly to print. For using the hpfhdl utility, do the following:

1. In your design directory, use a text editor to create a header file with information on the plotter, the font, the scale, and the file format. These header fields are described in Header File Format.

   Sample Header File

   ```
   hp7580
   1 vector_font
   B 500
   B
   <path to project_dir>/worklib
   <your_install_dir>/share/library/standard
   <your_install_dir>/share/library/lsttl
   ```

   Note: There must be no extra lines in the header file after the last directory path for hpfhdl to work properly.

2. Identify the name of the binary drawing file that you want to plot.

   Each drawing directory contains a drawing file in the binary format. The filename is pageN.csb. For example, if you want to plot the first page of a schematic mylogic, the binary file for the schematic page is page1.csb, which is located at:

/<project_dir>/worklib/mylogic/sch_1

In this example, we will plot the binary file /<project_dir>/worklib/mylogic/sch_1/page1.csb.

3. Enter the hpfhdl command with parameters as follows:

   - To create a plot file named myoutputfile to print later:

```
hpfhdl -f myoutputfile -2 myheadername.header worklib/mylogic/sch_1/page1.csb
```

     where myheadername.header is the name of the header file.

   - To print directly to the plotter specified in the header file:

```
hpfhdl -2 myheadername.header worklib/mylogic/sch_1/page1.csb
```

Header File Format

The input file that hpfhdl reads has a special header containing information about the type of the plotter, the line width, the scale, and the format of the graphical information. If the file is binary, the header includes a list of paths to the directories where the hpfhdl utility searches to find the referenced symbols in the binary file for the drawing.

Header fields and options are

- Plotter name

  Any plotter name in the .cdsplotinit file

- Line weight
  - NORMAL_WEIGHT lines (1)
  - HEAVY_WEIGHT lines (2)
- Font type

  The font type parameter appears on the same line as the line weight parameter. Font type is optional. If you do not specify a font type, the vector_font type is used for plotting. The following font types are available:

  - vector_font (default)
  - cursive_font
  - valid_font
  - greek_font
  - milspec_font

- `symbol_font`
- `gothic_font`
- `native_font` (defaults to Courier)
- `Courier`
- `Times New Roman`
- `Helvetica`

Important

If you use resident (native) fonts, the `.cdsplotinit` file must include the following entry:

```
:residentFonts:
```

- Scale

  The scale can be

  - An ASCII positive real number string
  - Any paper size specified for the plotter in the `.cdsplotinit` file

- Coordinates-per-inch

  These are the plotter coordinates and they appear on the same line as the scale parameter. Coordinates-per-inch is an ASCII integer; `hpfhdl` multiplies the incoming coordinates by the scale to get correct plotter coordinates.

  Concept HDL uses the following plotter coordinates:

Coordinates-Per-Inch	For Plot Type
500	Decimal
508	Metric
400	Fractional

- Encoding Type

  An encoding type can be

  V Vectorized

B Binary

C Component or symbol

An illegal encoding type causes the `hpfhdl` program to terminate.

- Directory Paths to Libraries

  A list of directory paths follows the encoding type only if the encoding type is binary (B).

  Specify the absolute directory paths to all the libraries used in your design. The directory paths are used to find the referenced symbols in the binary file for the drawing. For example, if you have used the `MERGE` symbol from the `standard` library in your schematic page, the `MERGE` symbol is referenced in the binary file for the schematic page. The `hpfhdl` utility will be able to plot the MERGE symbol in the schematic only if you have given the path to the `standard` library in the header file as below:

  ```
  <your_install_dir>/share/library/standard
  ```

  If you do not give the path to the `standard` library in the header file, `hpfhdl` displays the following error message when you plot the schematic page and the MERGE symbol will not be plotted correctly.

  ```
  Error locating body 'MERGE' version 1
  ```

  Note: Do not use environment variables to specify the directory path to a library. For example, if you use the `$INST_DIR` environment variable to point to the installation directory for Cadence tools, do not specify the path to the `standard` library in the header file as below:

  ```
  $INST_DIR/share/library/standard
  ```

Caution

There must be no extra lines in the header file after the last directory path or hpfhdl will not work properly.

The following table shows three sample headers.

Headers	Comments
hp7580	HP 7580 pen plotter
1 milspec_font	NORMAL lines, Milspec font
D 500	scaled to D size, 500 units per inch
V	vectorized format file

Headers	Comments
vers11	11-inch Versatec plotter
2 gothic_font	HEAVY lines, Gothic font
B 500	scaled to B size, 500 units per inch
B	binary format file
<path to project_dir>/worklib <your_install_dir>/share/library/standard <your_install_dir>/share/library/lsttl	directories to search for drawings
calcomp1043	CalComp 1043 pen plotter
1	NORMAL lines, default font
E 500	scaled to E size, 500 units per inch
B	binary format file

Note: You must not include comments in a header file. Comments are included in these examples for your information only.

hpfhdl Command Syntax

```
hpfhdl [-f|-v outputfile] [-o] [-p papersize] [-2 <headerfile> <path_to_drawing>]
```

-f	Writes the data to a new version of the output file
-v	Writes a vector format file to a new version of the output file
outputfile	If no output file is specified, the output is sent to the printer or the plotter
-o	This parameter operates if you specify the -f option, which implies that `hpfhdl` overwrites the new version of the output file (-f specification) rather than append data to the file specified, which is the default behavior.
-p	Specifies the paper size. This value must already be defined for the plotter in the `.cdsplotinit` file. The default is the first paper size entry defined for the plotter in the `.cdsplotinit` file.
headerfile	The name of the header file. For more information, see Header File Format

path to drawing	The path to the drawing you want to plot. For example, if you want to plot the `page1.csb` file at `/<project_dir>/worklib/mylogic/sch_1/`, the `hpfhdl` command line will be as below: `hpfhdl -f myoutputfile -2 myheadername.header worklib/mylogic/sch_1/page1.csb` where `myheadername.header` is the name of the header file.

Hierarchical Plotting

Hierarchical plotting in Concept HDL allows you to selectively plot the schematics of cells that belong to a hierarchy of the design. Hierarchical Plotting is available in both Windows Plotting and HPF Plotting. For details on Windows Plotting, refer to Windows Plotting on Windows and UNIX Platforms on page 387. For details on Hierarchical Plotting, refer to HPF Plotting on UNIX Platforms on page 402.

The following topics are discussed in this section:

- Hierarchical Plotting in Hierarchy, Expanded, and Occurrence Edit Modes on page 424
- Changing the Order in Which Designs Are Plotted on page 426
- Plotting Hierarchical Designs on page 429

Important

If you open a design that was created in Concept HDL 14.0 in Concept HDL 14.2 and perform module ordering or hierarchical plotting (in Windows or HPF Plotting), the `module_order.dat` file (located at `<worklib>/<root design/sch_1`) is written in a new format. If you open the design again in Concept HDL 14.0, you cannot see the module ordering information in Concept HDL 14.0. Cadence recommends that you maintain a copy of the `module_order.dat` file created by Concept HDL 14.0 or maintain a copy of the design so that you do not lose the module ordering information in Concept HDL 14.0. For more information, see Changes in Module Ordering from Concept HDL 14.0 to Concept HDL 14.2 on page 352.

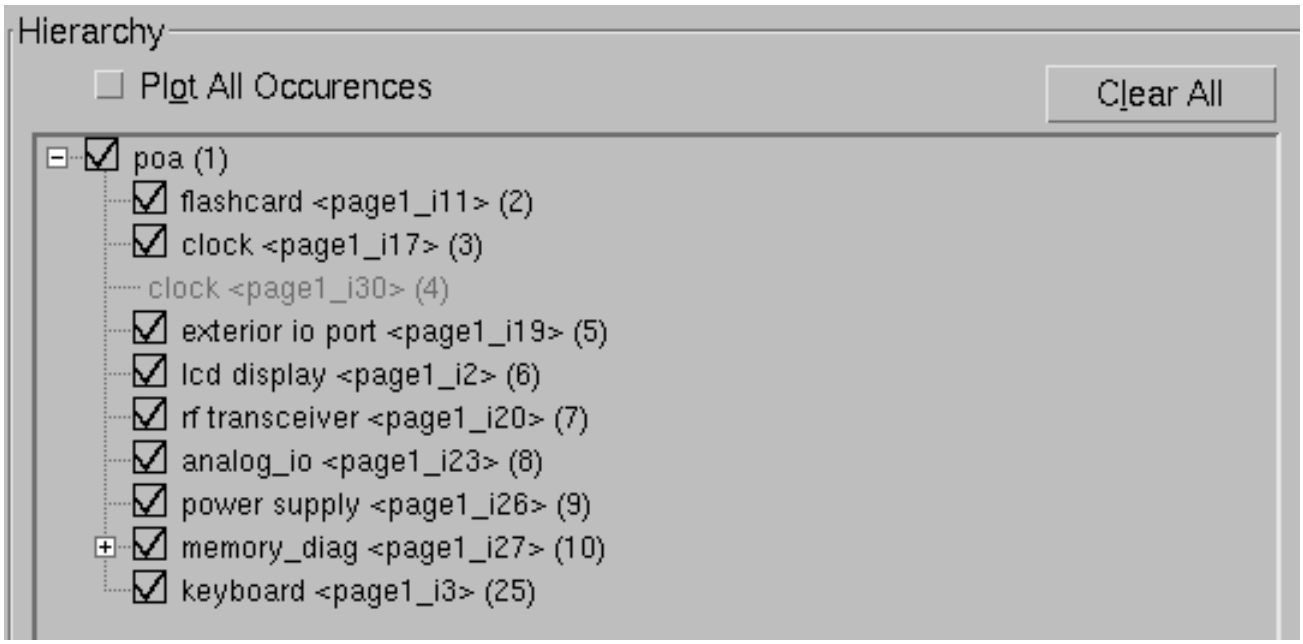
Hierarchical Plotting in Hierarchy, Expanded, and Occurrence Edit Modes

Hierarchical plotting is supported in Hierarchy, Expanded, and Occurrence Edit modes. In the Occurrence Edit mode, occurrence properties are also plotted. When you plot a hierarchical design with multiple instantiations of a block in the Occurrence Edit mode, Concept HDL plots

every occurrence of the block. When you plot the same design in the Hierarchy mode or Expanded mode, Concept HDL plots the block only once.

The following figure shows how the hierarchy is displayed in the Hierarchy and Expanded modes:

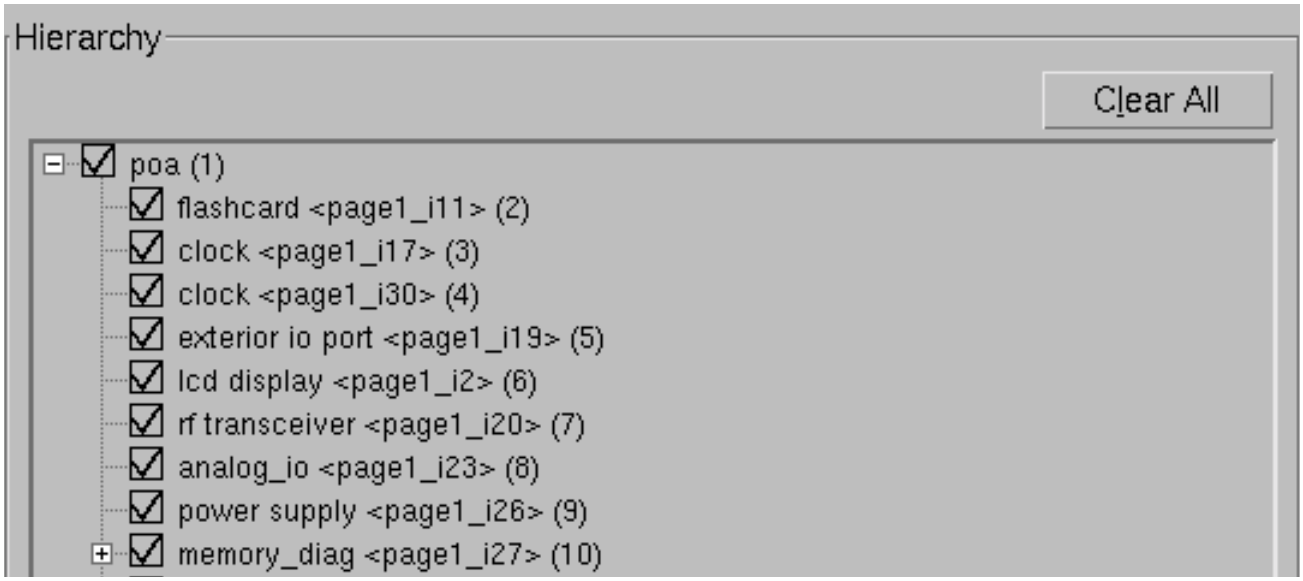
Hierarchy in the Hierarchy and Expanded Modes



Notice that two instances of the block CLOCK is displayed in the hierarchy. This means that the block CLOCK is instantiated twice in the design. Note that the second instance of the block is grayed out. This is because, in the Hierarchy or Expanded mode, only one occurrence of a block is plotted by default. Select the *Plot All Occurrences* check box if you want to plot all occurrences of a block.

The following figure shows how the hierarchy is displayed in the Occurrence Edit mode:

Hierarchy in Occurrence Edit Mode



In the Occurrence Edit mode, both the occurrences of the block CLOCK are enabled. This is because, when you plot in the Occurrence Edit mode, Concept HDL plots each occurrence of a block. The occurrence properties specified on each instance of a block are also plotted.

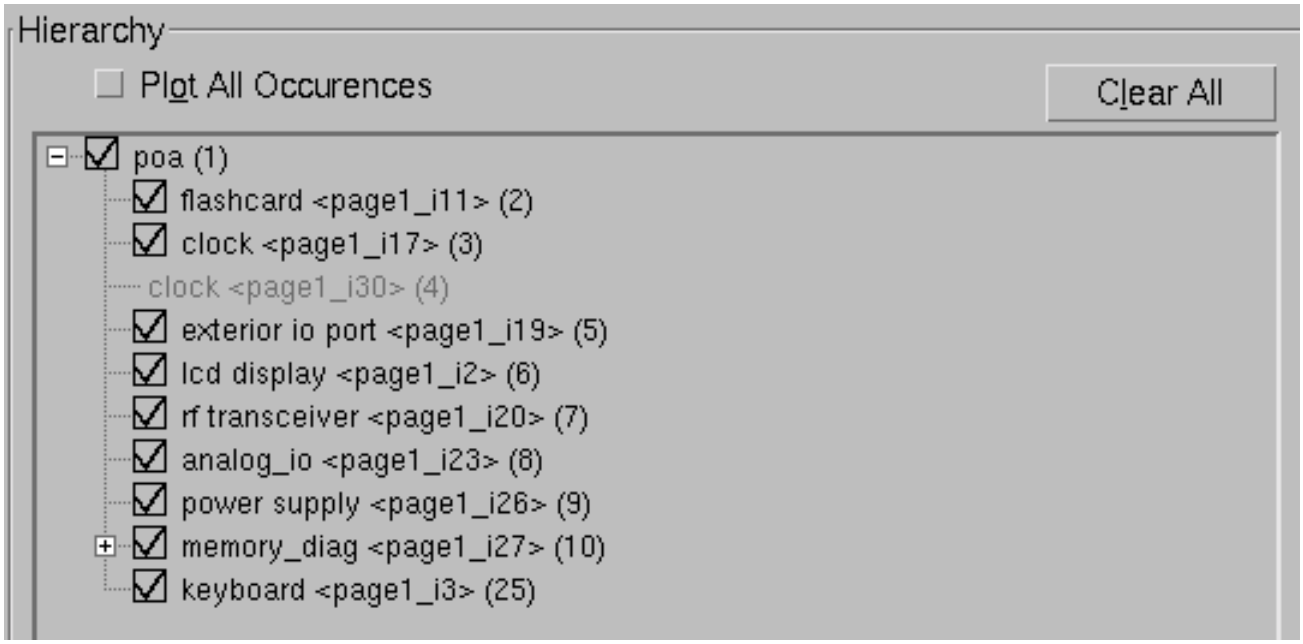
For more information on working in the Hierarchy, Expanded, or Occurrence Edit mode, see Modes in Concept HDL on page 113.

Changing the Order in Which Designs Are Plotted

You can modify the order in which designs are plotted. To do this, you must perform module ordering before plotting the design. If you exclude a module during module ordering, that module will not be displayed in the hierarchy. For more information on module ordering, see Module Ordering on page 351.

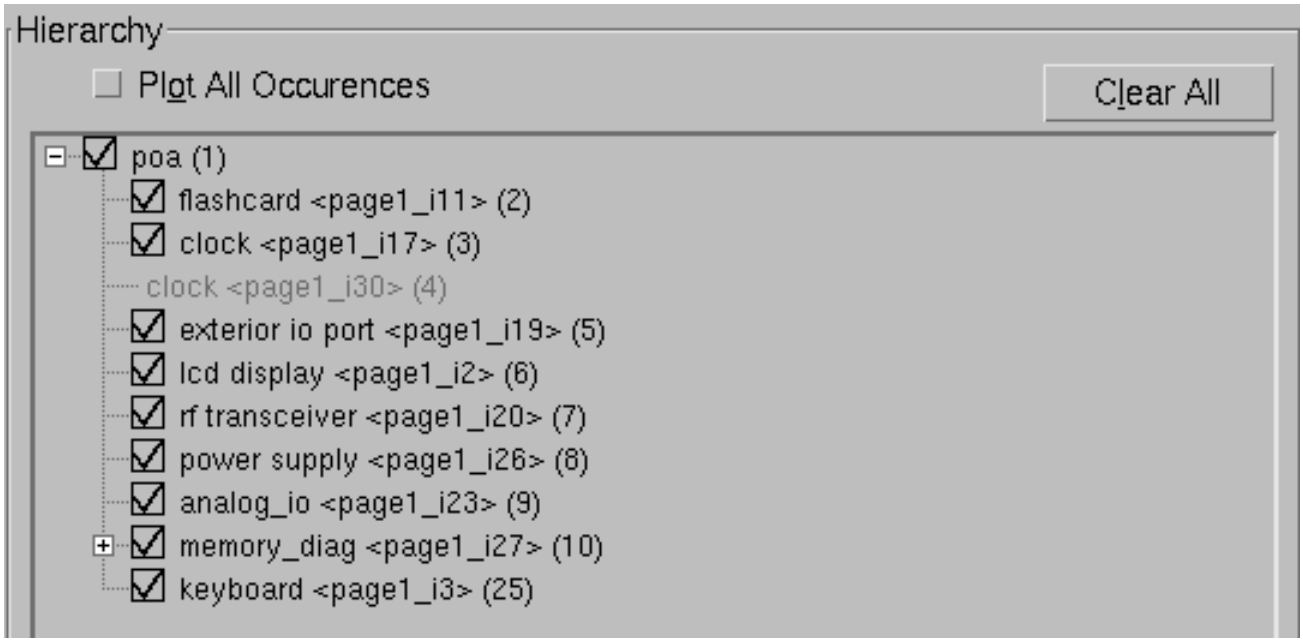
Let’s take the following example to see how module ordering impacts hierarchical plotting:

Figure 16-1 Hierarchy Before Module Ordering



If you want to plot the `analog_io` design after the `power supply` design, perform module ordering to move the `analog_io` module after the `power supply` module. This is how the hierarchy will be displayed after you have performed module ordering:

Figure 16-2 Hierarchy After Module Ordering

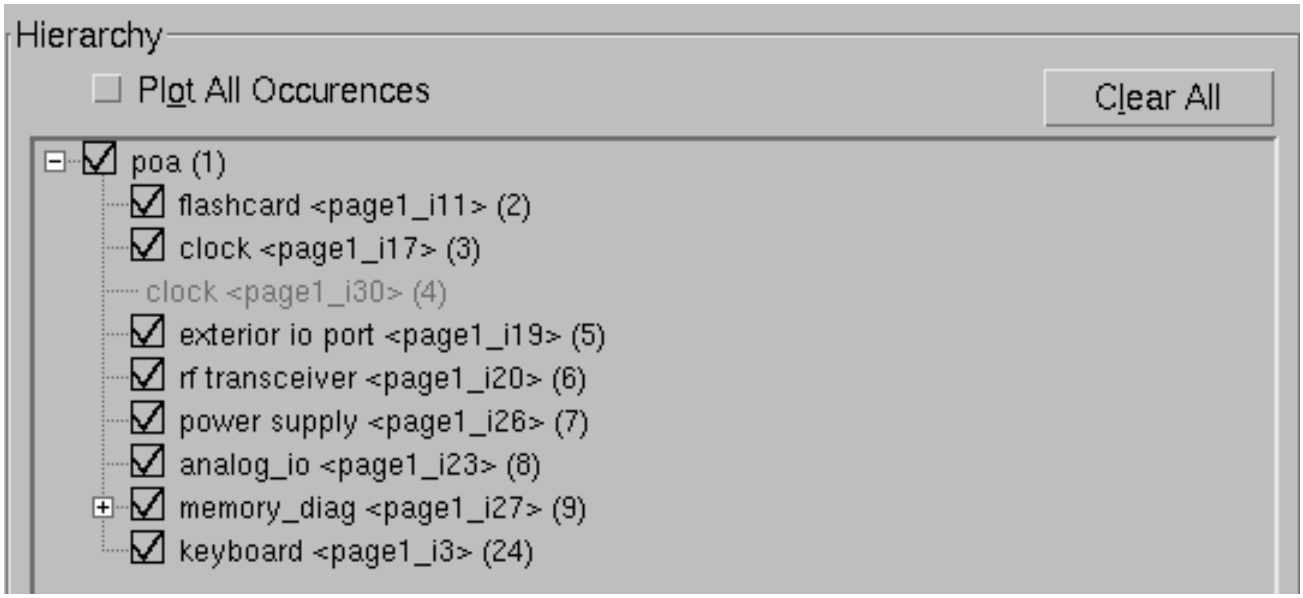


Notice that the schematic page numbers have also changed after module ordering. In Figure 16-1 on page 427, the `analog_io` design had the page number 8 and the `power supply` design had the page number 9. After module ordering, the page number has changed to 8 for the `power supply` design and to 9 for the `analog_io` design. For more information on page numbering, see Displaying and Working with Schematic Page Numbers on page 358.

Note: If you perform page renumbering, the order in which the designs are plotted will not change. If you want the order in which the designs are plotted to change after you do page renumbering, you must perform module ordering. For more information, see Page Renumbering, Module Ordering and Hierarchical Plotting on page 368.

If you do not want the `lcd display` design to be plotted, you can clear the check box next to the design name or exclude the `lcd display` module by performing module ordering. If you exclude the `lcd display` module during module ordering, the `lcd display` design will not be displayed in the hierarchy. This is how the hierarchy will look after you exclude the `lcd display` module by performing module ordering.

Figure 16-3 Hierarchy After Excluding the lcd display Module



Note that the `lcd display` design is no longer displayed in the hierarchy. The page numbers of the designs have also changed.

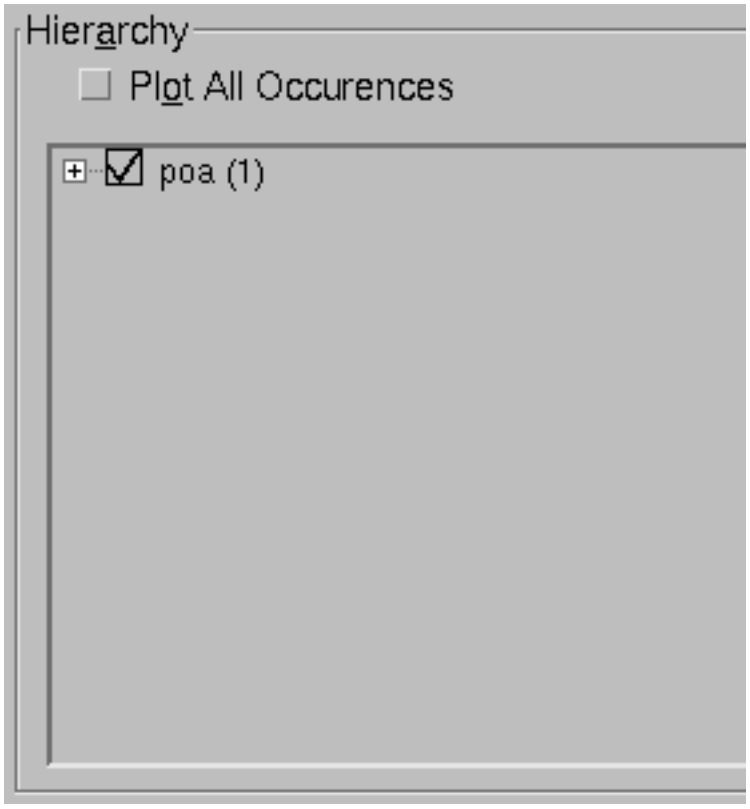
Plotting Hierarchical Designs

Hierarchical plotting is supported in Hierarchy, Expanded, and Occurrence Edit modes. If you want to plot occurrence properties, you must change to the Occurrence Edit mode before plotting the schematic. For more information on working in the Occurrence Edit mode, see Occurrence Edit Mode on page 114.

1. Choose *File > Save All*.
2. Choose *File > Plot*.

   The *Plot* dialog box appears if you are performing Windows Plotting. The *HPF Plot* dialog box appears if you are performing HPF Plotting on UNIX.
3. Select *Hierarchy* to extend the *Plot* dialog box.

   Concept HDL displays the hierarchical structure of the root design.



The root design is displayed as `poa (1)`, where

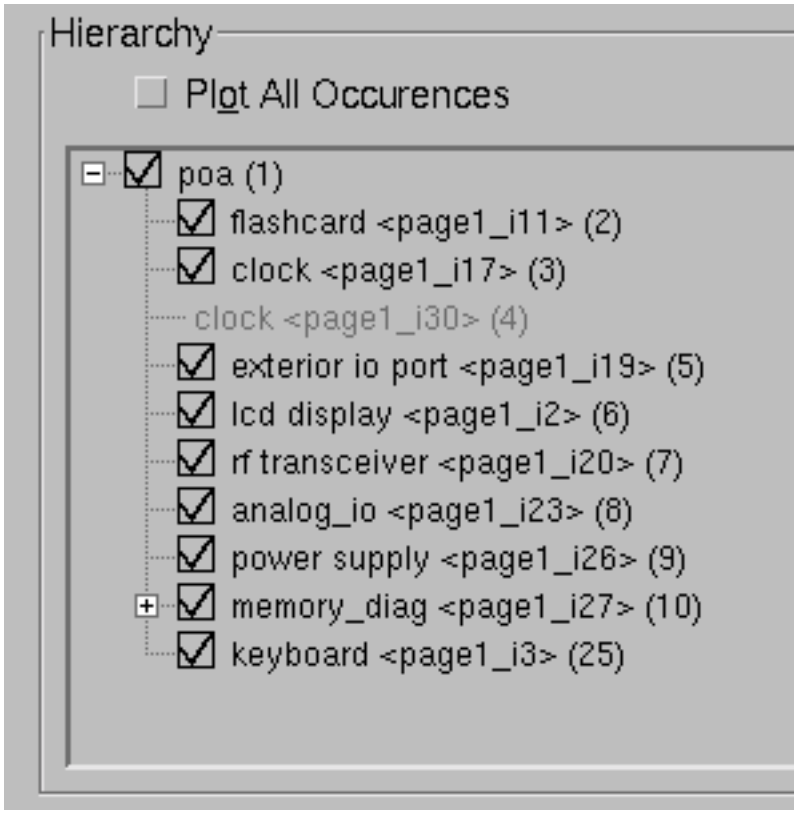
- `poa` is the name of the root design, and
- 1 is the number of the page in which the `poa` design will be plotted.

If the root design had three pages, the design name will be displayed as `poa (1-3)`.

Note: The page number will not be plotted by default. If you want the page number to be plotted, you must use the `CURRENT_DESIGN_SHEET` custom text variable on the page in the design. For more information on page numbering, see Displaying and Working with Schematic Page Numbers on page 358.

The check box in a white background indicates that the design called `poa` and all sub designs under it will be plotted.

4. Click the + icon next to `poa` for Concept HDL to display all sub designs in it.



The first sub design is displayed as `flashcard <page1_i11> (2)`, where

- `flashcard` is the name of the sub design,
- `<page1_i11>` indicates that the sub design is instantiated as instance `i11` on page 1 of the root design, and
- 2 is the number of the page in which the `flashcard` sub design will be plotted.

  If the `flashcard` sub design had two pages, the design name will be displayed as `flashcard <page1_i11> (2-3)`.

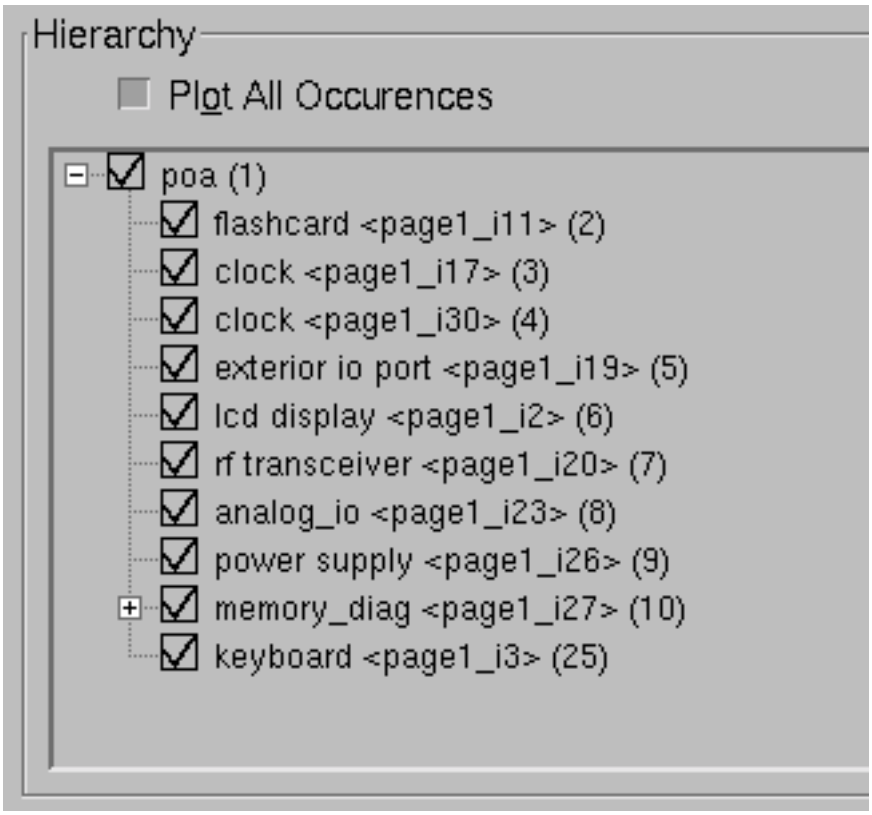
Note: The page number will not be plotted by default. If you want the page number to be plotted, you must use the `CURRENT_DESIGN_SHEET` custom text variable on the pages in the design. For more information on page numbering, see Displaying and Working with Schematic Page Numbers on page 358.

Concept HDL displays the check boxes next to all sub designs under the design `poa` as selected. This means that all the sub designs will be plotted.

In the Hierarchy or Expanded mode, only one occurrence of a block is plotted by default. There are two occurrences of the block `clock` in the above figure. The second instance of the block `clock` will be grayed out in the Hierarchy or Expanded mode. In the Occurrence Edit mode, all occurrences of the block `clock` will be plotted by default. For more information, see Hierarchical Plotting in Hierarchy, Expanded, and Occurrence Edit Modes on page 424.

5. Select the *Plot All Occurrences* check box if you want to plot both the occurrences of the block `clock` in the design.

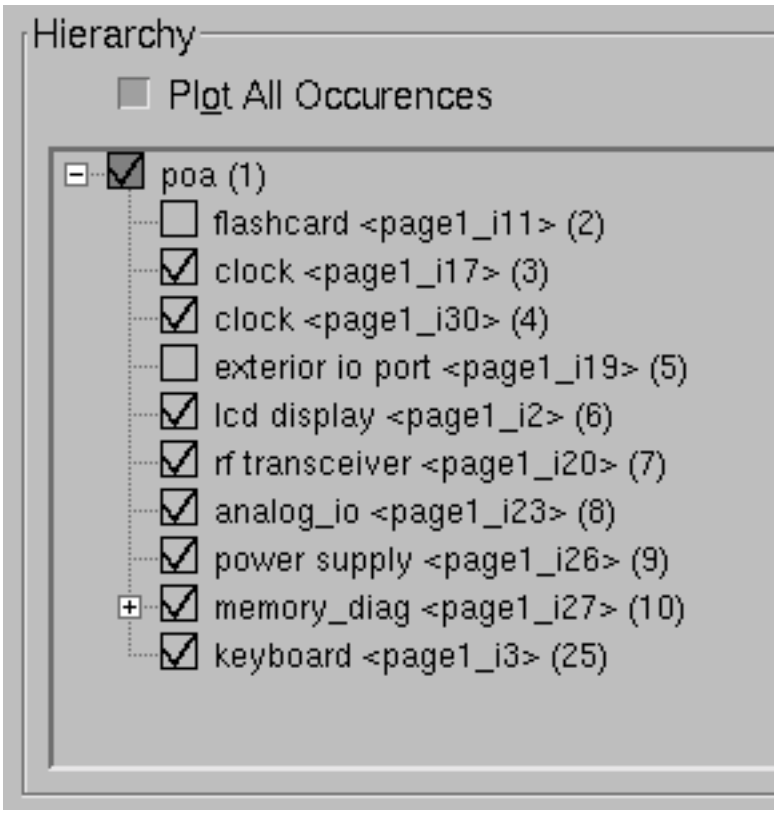
   Note: This check box is displayed only if you are in the Hierarchy or Expanded mode.



Both the occurrences of the block `clock` are selected for plotting.

You can select or deselect designs for plotting. To modify the order in which the designs are plotted, you must perform module ordering before plotting the design. For more information, see Changing the Order in Which Designs Are Plotted on page 426.

6. If you do not want to plot a sub design, clear the check box next to the sub design name.

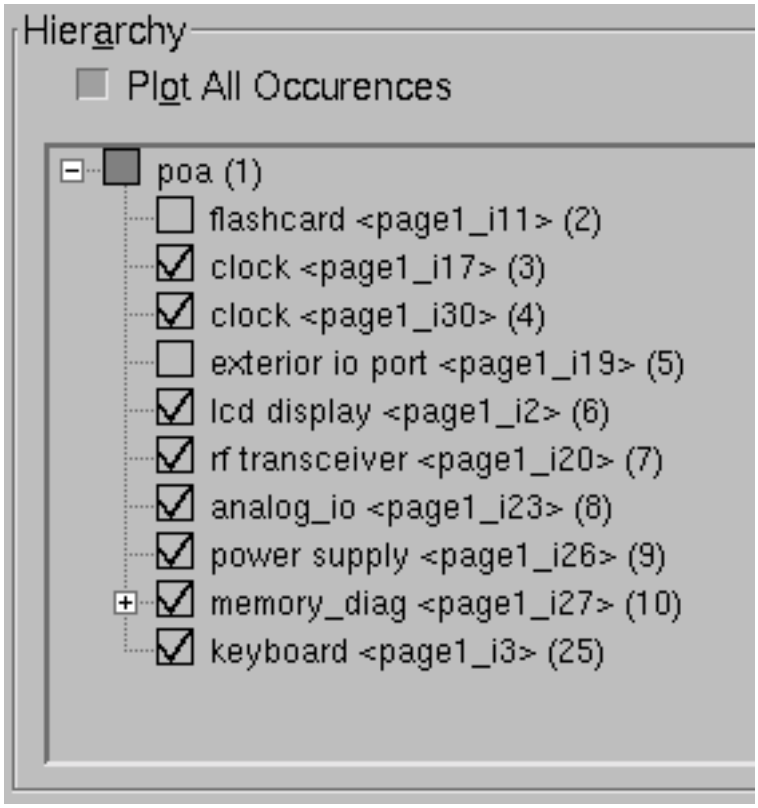


When you deselect some of the sub designs, Concept HDL displays the check box next to the design poa in a grey background. This indicates that Concept HDL will plot poa and only some sub designs in it.

To select a design for plotting, select the check box next to the design name.

Note: You can also click a design name to select or deselect the design for plotting. Each click on a design name selects or deselects the check box next to the design name.

7. Clear the check box next to poa.



The check box next to poa is displayed with a grey background. This indicates that Concept HDL will plot some sub designs in the design poa, but not poa. Even if you select all sub designs, Concept HDL will plot all the sub designs but not poa.

Customization of Plotting on UNIX Platforms

This section describes the customization of Windows and HPF plotting on UNIX platforms.

- Customization of Windows Plotting on UNIX Platforms on page 433
- Customization of HPF Plotting on UNIX Platforms on page 436

Customization of Windows Plotting on UNIX Platforms

Windows plotting can be customized on UNIX using the win.ini file.

`win.ini` is a text file that contains the plotter settings for the Windows plotting utility on UNIX. You can edit it to specify more plotters other than the default plotter. The `win.ini` file is located in the windows directory in your home area.

Caution

Errors caused while editing the win.ini file may change the behavior of Concept HDL.

To change the plotters displayed in the `Name` field of the Print dialog box.

Edit the `win.ini` file to add entries to the [DEVICES], [PRINTERPORTS], [PORTS], and [WINDOWS] sections.

Note: On UNIX, plotter-related sections like, PrinterPorts, ports and devices are usually not used because plotter information is taken from the registry. If you want these sections to control the plotters, you should set the environment variable `MWPRINT_USING_WIN_INI` to `true`.

[DEVICES] Section

The [devices] section names the active plotters. The settings in this section have the following syntax:

<device-name>=<driver-name, port-name>:

where <device-name> is the name of the device.

The <driver-name> is the driver filename (usually PSCRIPT for Postscript).

The <port-name> specifies the port to which the device is mapped (LPTn where n is any valid port number). This is only a logical port and does not indicate where the physical plotter is connected.

If a device is not connected currently, the <port-name> value should be the string specified in the NullPort settings (in the [windows] section).

Example

If you want to add a plotter say, "My plotter" which is a Postscript plotter then add the following line in the [devices] section:

```
[devices]
My plotter=PSCRIPT,LPT1:
```

[PRINTERPORTS] Section

This section is included for compatibility with applications based on earlier versions of Windows. It names all the plotters mentioned in the [Devices] section along with the postscript timeout values. The timeout values for a plotter are given in the user manual for the plotter.

The syntax of entries in this section is

```
<device-name>=<driver-name, port-name>:, <timeout-values>
```

The <timeout-values> must be taken from the plotter's handbook or mention 15,90.

Example

```
[PrinterPorts]
My Printer=PSCRIPT,LPT1:,15,90
```

where 15 and 90 are the Postscript timeout values.

[PORTS] Section

By default, Concept HDL will look for the PRINTER environment variable to determine the plotter for plotting. To override the PRINTER variable setting permanently, you can edit the [PORTS] section in the `win.ini` file.

Example

```
[PORTS]
LPT1:=lp -c -d<destination-printer> "%s"
```

[WINDOWS] Section

To display the new plotter you have added as the default selected plotter in the *Print* dialog box, you need to edit the [WINDOWS] section.

To set the plotter of your choice as the default selected plotter (from the plotters listed in the [DEVICES] and [PRINTERPORTS] sections), edit the [WINDOWS] section as follows:

```
[WINDOWS]
device=<device-name, driver-name, port-name>:
```

The device-name, driver-name, and port-name values must be specified in the [DEVICES] section.

Customization of HPF Plotting on UNIX Platforms

HPF plotting can be customized on UNIX using the .cdsplotinit file.

Before you plot a drawing, or create a plot file, you must specify the plotter you want to use and plotter-specific parameters in the .cdsplotinit file. This file can reside in any of the three locations given below. Concept HDL looks for the file in the following order:

1. Your home directory
2. The design directory
3. <your_install_dir>/tools/plot

While you can build a site-specific .cdsplotinit file using the files listed below, Cadence recommends using the interactive plotconfig utility located in <*your_install_dir*>/tools/plot/bin.

- For model file entries: <*your_install_dir*>/tools/plot/etc.
- For a description of the options used in .cdsplotinit and more example model entries, see <*your_install_dir*>/tools/plot/ samples/ cdsplotinit.sample.

The primary purpose for .cdsplotinit is the mapping of the plotter name (also menu name) to the plotter type and to the printcap queue, as shown in the following sample .cdsplotinit entry:

The following sample .cdsplotinit file contains entries for commonly used plotting devices.

```
vers11|v80: \
    :spool=lpr -Pvers11:\
    :query=lpq -Pvers11: \
    :remove=lprm -Pvers11 $3: \
    :manufacturer=Xerox Engineering Systems: \
    :type=intBW: \
    :maximumPages#10: \
    :resolution#200: \
    :compress: \
    :residentFonts: \
    :outtype=RASTER: \
    :instdir=/usr/valid: \
    :tmpdir=/usr/tmp: \
    :paperSize="B" 3200 2112:
postscript|Apple Laser Writer II NT/NTX: \
    :spool=lpr -Ppostscript: \
    :query=lpq -Ppostscript: \
    :remove=lprm -Ppostscript $3: \
    :residentFonts:\
    :manufacturer=Apple: \
```

```
    :type=postscript1: \
    :maximumPages#30: \
    :resolution#300: \
    :paperSize="A" 2400 3150 75 75:
postscript|Apple LaserWriter II NT/NTX: \
    :spool=lpr -Ppostscript: \
    :query=lpq -Ppostscript: \
    :remove=lprm -Ppostscript $3: \
    :residentFonts: \
    :manufacturer=Apple Computer: \
    :type=postscript1: \
    :maximumPages#30: \
    :resolution#300: \
    :paperSize="B" 3150 4950 75 75:
hp7580|Hewlett-Packard 7580A/B: \
    :spool=lpr -Php7580: \
    :query=lpq -Php7580: \
    :remove=lprm -Php7580 $3: \
    :manufacturer=Hewlett-Packard: \
    :type=hp7580: \
    :maximumPages#1: \
    :resolution#1016: \
    :white#8:black#1:red#6:yellow#4:green#7: \
    :cyan#2:blue#5:magenta#3: \
    :paperSize="A" 5276 8776:   \
    :paperSize="B" 13912 8776:  \
    :paperSize="C" 18992 14872: \
    :paperSize="D" 31184 19952: \
    :paperSize="A1" 30280 21360: \
    :paperSize="A2" 20400 14400: \
    :paperSize="A3" 13440 9480: \
    :paperSize="A4" 5040 9480:
```

For more information on the `.cdsplotinit` file, see the *Plotter Configuration User Guide*.

Frequently Asked Questions in Plotting

This section contains the answers to most frequently asked questions about plotting in Concept HDL. To view the answer to any question, click on that question in the list below.

Which are the plotters supported in HPF Plotting?

On Windows NT, how do I select a plotter that is on the network?

Can I change the thickness or font of the text on a schematic?

In HPF plotting, how can I plot a drawing so that it fits in a paper size that is smaller than the size of the drawing.

In HPF plotting, how can I plot a drawing so that it is scaled to fit in a paper size that is larger than the size of the drawing.

How do I select paper sizes in HPF plotting?

Can I exclude some design modules from plotting and cross referencing?

Do I have to set any environment variable to run the hpfhdl command from a UNIX terminal?

Do set console commands affect plotting through the Plot dialog box?

How can I plot a colored schematic on color plotters?

How can I view a spool file created by plotting?

How can I plot from the command line on UNIX?

How can I create a PDF output file of a Concept HDL schematic?

How can I plot a schematic for which I have only read-only permissions?

How do I plot all the pages in a flat schematic at the same time?

Can I plot hierarchical schematics?

Can I plot occurrence properties?

Is previewing supported for HPF plotting?

In Windows plotting mode, can I preview all the plot pages together?

From where can I select different setup options?

Which are the plotters supported in HPF Plotting?

Click the *Cadence Plotting Services* link in the SourceLink (http://sourcelink.cadence.com) main page for information on the list of plotters that are supported for HPF plotting.

On Windows NT, how do I select a plotter that is on the network?

In Concept HDL, do the following:

1. Choose *File > Plot Setup*.

   The *Plotting* tab in the *Concept Options* dialog box is displayed.

2. Click *Setup*.

   The *Print Setup* dialog box appears.

3. Click *Network*

4. Select the printer and click *OK*.

Can I change the thickness or font of the text on a schematic?

In the Windows plotting mode, the thickness of the text is always the same as the thickness of a thin line. You can adjust this thickness to increase or decrease the text thickness also.

1. Access the *Plotting* (Windows) tab of the *Concept Options* dialog box.
2. Increase or decrease the size specified in the *Single Line Width* field.

   The thickness of the text changes accordingly.

Note: You cannot change the font of the text in Windows plotting mode.

If you are using HPF plotting, do the following to change the thickness of the text.

1. Access the *Plotting* (HPF) tab of the *Concept Options* dialog box.
2. Increase or decrease the scale factor specified in the *Wire Scale Factor* field.

   The thickness of the text changes accordingly.

In the HPF plotting mode, you can use one of the following fonts for the text.

- VECTOR
- CURSIVE
- NATIVE
- GOTHIC
- SYMBOL
- GREEK
- VALID
- MILSPEC

For more information, see HPF Plotting on UNIX Platforms on page 402.

In HPF plotting, how can I plot a drawing so that it fits in a paper size that is smaller than the size of the drawing.

For example, you have used the `C SIZE PAGE` page border (17 x 22 inch) symbol in your drawing and want to plot the drawing so that it fits in a single sheet of paper size `A` (8 1/2 x 11 inch).

Do the following:

1. Access the *Plotting* (HPF) tab of the *Concept Options* dialog box.
2. Specify `A` in the *Specify Page Size* field and choose `A` in the *Scale to Page Size* drop-down list.

When you plot the drawing the entire drawing is plotted on a single sheet of `A` size paper.

Concept HDL allows you to scale a drawing to one of the five standard page sizes A, B, C, D, or E. In addition to the standard page sizes, you can define your own custom page sizes for the plotter in the `.cdsplotinit` file. However, you can scale a drawing to plot in a custom page size only with the `hardcopy` console command. For example, the command:

```
hardcopy MYPAGE
```

scales the currently open drawing to plot in page size `MYPAGE` (the custom page size you have defined in `.cdsplotinit` file).

In HPF plotting, how can I plot a drawing so that it is scaled to fit in a paper size that is larger than the size of the drawing.

For example, you have used the `A SIZE PAGE` page border (8 1/2 x 11 inch) symbol in your drawing and want to plot the drawing so that it is scaled to fit the complete area of paper size `C` paper size (17 x 22 inch).

Do the following:

1. Access the *Plotting* (HPF) tab of the *Concept Options* dialog box.
2. Specify `C` in the *Specify Page Size* field and choose `C` in the *Scale to Page Size* drop-down list.

When you plot the drawing the entire drawing is plotted to fit the complete area of a single sheet of `C` size paper.

Concept HDL allows you to scale a drawing to one of the five standard page sizes A, B, C, D, or E. In addition to the standard page sizes, you can define your own custom page sizes for

the plotter in the .cdsplotinit file. However, you can scale a drawing to plot in a custom page size only with the hardcopy console command. For example, the command:

```
hardcopy MYPAGE
```

scales the currently open drawing to plot in page size MYPAGE (the custom page size you have defined in the .cdsplotinit file).

How do I select paper sizes in HPF plotting?

For each plotter in the .cdsplotinit file, a number of paper sizes along with their dimensions are defined. You can specify any of these paper sizes in the *Specify Page Size* field of the *Plotting* (HPF) tab of the *Concept Options* dialog box.

If you are using the hardcopy console command, you can use the set papersize <size> console command to specify the paper size.

Note: If an invalid paper size is specified, the first paper size defined for the plotter will be used.

Can I exclude some design modules from plotting and cross referencing?

Yes, in Hierarchical plotting you can choose to exclude some modules from plotting through the Module Order dialog box or the xmodules.dat file. For more information on module ordering see Module Ordering on page 351.

Do I have to set any environment variable to run the hpfhdl command from a UNIX terminal?

Yes. You have to set an environment variable to run the hpfhdl command from a UNIX terminal as below:

On Sun Solaris

Set the LD_LIBRARY_PATH environment variable as below:

```
setenv LD_LIBRARY_PATH <your_install_dir>/tools/editor/lib $LD_LIBRARY_PATH
```

On IBM AIX

Set the LIBPATH environment variable as below:

```
setenv LIBPATH <your_install_dir>/tools/editor/lib $LIBPATH
```

On HP-UX

Set the SHLIB_PATH environment variable as below:

```
setenv SHLIB_PATH <your_install_dir>/tools/editor/lib $SHLIB_PATH
```

Do set console commands affect plotting through the Plot dialog box?

No. The `set` console commands affect plotting through the `plot` console command only. They do not have any effect on plotting through the *Plot* dialog box.

The `set` console commands do not change any directives in the `.cpm` file and are meant only for the current Concept HDL session. So, any change made through them is not visible in the dialog box settings.

How can I plot a colored schematic on color plotters?

Do the following:

1. Access the *Plotting* (Windows) tab of the *Concept Options* dialog box.
2. Select the Color option.

How can I view a spool file created by plotting?

You can use the following third-party freeware utilities that are available on the Internet to view spool files:

- GSView for Windows
- Ghostview for SUN Solaris, IBM AIX and HP-UX
- Pageview for Sun Solaris.

How can I plot from the command line on UNIX?

On UNIX, use the `hpfhdl` command or the `nconcepthdl` command (the command that allows you to run Concept HDL scripts in non-graphical mode) to plot from the command line.

To use the `nconcepthdl` command, do the following:

1. Enter the `hardcopy` commands for the drawings that you want to plot in a script file, say `myplot.scr`.
2. Use the following command to plot from the command line:

```
nconcepthdl -proj <project_name>.cpm -scr myplot.scr
```

Note: The `nconcepthdl` and `hpfhdl` commands do not allow you to plot occurrence properties from the command line. This is because it plots directly from the binary files for the drawing.

For more information on the `nconcepthdl` command, see Nongraphical Concept HDL (nconcepthdl) on page 611. See also, Do I have to set any environment variable to run the hpfhdl command from a UNIX terminal? on page 441.

How can I create a PDF output file of a Concept HDL schematic?

On Windows NT

To create a PDF output file of a Concept HDL schematic on Windows NT, you must have the Adobe Acrobat software from Adobe Systems Inc., installed on your machine.

In Concept HDL, do the following:

1. Choose *File > Plot*.

   The *Plot* dialog box appears.

2. Select *Acrobat PDFWriter* or *Acrobat Distiller* in the *Printer Name* drop-down list.

3. Click *Plot*.

   The *File Save As* dialog box appears.

4. Specify the PDF file name and click *Save*.

   This generates the PDF output file for your Concept HDL schematic.

On Unix

To create a PDF output file of a Concept HDL schematic on UNIX, you must have the Adobe Acrobat Distiller software from Adobe Systems Inc., installed on your machine.

If you are using HPF plotting on UNIX, do the following:

1. Access the *Plotting* (HPF) tab of the *Concept Options* dialog box.

2. Select the *Batch Plot* check box.

   The drawing will be plotted to a file `vw.spool` file if this check box is selected.

3. Click *OK*.
4. Choose *File > Plot*.

   The *Hard-Copy Plot Facility* dialog box appears.
5. Click *OK*.

   The `vw.spool` file is created in the project directory.
6. Open a UNIX terminal.
7. Change to the project directory.

   The project directory is the directory that has the `<project_name>.cpm` file for your project.
8. Enter the following command in the UNIX terminal:

   ```
   distill vw.spool
   ```

   The `vw.spool` file is processed and a `vw.spool.pdf` file is created in the project directory.

   Note: `distill` is the name of the executable for Adobe Acrobat Distiller. For more information on the options for the `distill` command, use the following command:

   ```
   distill -help command.
   ```

If you are using Windows plotting on UNIX, do the following:

1. In Concept HDL, choose *File > Plot*.

   The *Plot* dialog box appears.
2. Select a postscript plotter from the *Printer Name* drop-down list.
3. Click the *Properties* button next to the *Printer Name* drop-down list.

   The *Options* dialog box appears.
4. Select the *Encapsulated Postscript File* or *File* check box and enter the name of the file, say `myplot.ps`, in the *Name* field.
5. Click *OK*.

   The *Plot* dialog box appears.
6. Click *Plot*.

   The `myplot.ps` file is created in the project directory.

7. Open a UNIX terminal.
8. Change to the project directory.

   The project directory is the directory that has the `<project_name>.cpm` file for your project.
9. Enter the following command in the UNIX terminal:

   ```
   distill myplot.ps
   ```

   The `myplot.ps` file is processed and a `myplot.pdf` file is created in the project directory.

   Note: `distill` is the name of the executable for Adobe Acrobat Distiller. For more information on the options for the `distill` command, use the following command:

   ```
   distill -help command.
   ```

How can I plot a schematic for which I have only read-only permissions?

Windows Plotting Mode

On both Windows NT and UNIX, plotting is transparent. You only need to ensure that you have write permissions in the directory for the output file.

- If you are plotting through the *Plot* dialog box, you are prompted to select the directory and the file name for the output file. Select a directory in which you have write permissions.
- If you are plotting through the `plot` console command, you can specify the directory for the output file using the `set wplot_file` command as below:

  ```
  set wplot_file <path>/<filename>.
  ```

  Ensure that you have write permissions in the directory for the output file.

  Note: If the plotter name consists of any special characters, precede the first such character with a \ (backslash character) and put the plotter name within quotation marks.

HPF plotting Mode

In HPF plotting, the spool file `vw.spool` is created in a temporary directory (/tmp) of the system from where Concept HDL was launched.

To create the spool file in another location, you need to set the environment variable `CDS_HPF_TMP` to a directory where you have write permissions.

Note: The spool file `vw.spool` is created in the directory specified using the `CDS_HPF_TMP` environment variable only if you are plotting a schematic for which you have read-only permissions. If you set this environment variable and plot a schematic for which you have write permissions, the spool file `vw.spool` is created in the project directory.

How do I plot all the pages in a flat schematic at the same time?

You can use wild cards in both the `plot` and `hardcopy` console commands. Please refer to sections Plot Command and Using the hardcopy Command for details.

- If you are using Windows plotting, the following console command plots all the pages of the flat schematic `mydesign.sch.1`:

  ```
  plot mydesign.sch.1.*
  ```

- If you are using HPF plotting, the following console command plots all the pages of the flat schematic `mydesign.sch.1`:

  ```
  ha a mydesign.sch.1.*
  ```

Can I plot hierarchical schematics?

Yes, you can plot hierarchical schematics in Windows plotting mode on both Windows NT and UNIX and in HPF plotting mode on UNIX. Refer to Hierarchical Plotting for more details.

Can I plot occurrence properties?

Yes, you can take plots of occurrence properties on a schematic using both Windows plotting and HPF plotting.

To plot occurrence properties, you have to switch to the Occurrence Edit mode in Concept HDL.

Note: In the Windows plotting mode, you can plot occurrence properties of any drawing in the design. However, in the HPF plotting mode you can plot only the occurrence properties of the currently open drawing.

Is previewing supported for HPF plotting?

Previewing is supported only in Windows plotting mode on both Windows NT and UNIX platforms. It is not supported in the HPF plotting mode.

In Windows plotting mode, can I preview all the plot pages together?

No, you can only preview the currently open page of the schematic.

You see multiple pages in the preview if you have selected the *Adjust to % Normal Size* option instead of the *Fit to Page* option in the *Plotting* (Windows) tab of the *Concept Options* dialog box. If you have selected the *Adjust to % Normal Size* option, the schematic page may span a few plotter papers. All those pages are shown in the preview also. For example, if the schematic page spans four plotter papers, you can view four pages in the preview.

From where can I select different setup options?

You can select different setup options through the *Plot Setup* dialog box. This is available through the *File > Plot Setup* or the *Concept Options* menu. In the *Plot Setup* dialog box and the *Plot* dialog box, the *Properties* button invokes a dialog box showing the system level plot setup options.

It is not recommended to use *Properties* dialog box for setup options. This dialog box should only be used on UNIX for setting the *Print to File* option.

17

Cross-Referencing Your Design

This chapter includes the following:

- Overview on page 449
- About Cross References on page 450
- How CRefer Cross-References a Design on page 455
- Getting Started with CRefer on page 461
- Using CRefer on page 468
- Understanding CRefer Output on page 479
- Reference Information on page 483

Overview

When you view a plot of a schematic, it is often difficult to trace a signal. The CRefer tool traces the signals in a schematic drawing and annotates their locations. The annotations by CRefer are called cross references.

CRefer places the signal cross-references next to each signal and creates schematic reports that contain the list of signal and part cross references.

Depending upon the nature of a signal, CRefer performs the following tasks:

- For each output signal in the schematic, CRefer lists all input locations where that signal appears.
- For each input signal, CRefer lists the locations of the sources of that signal.
- For each interface signal in a design, CRefer lists all the nets that are connected to it across the hierarchy.

Besides creating schematic reports, CRefer also creates text reports that contain cross referencing information about signals and parts in a design.

About Cross References

Types of Cross References

CRefer places two types of cross-references, flat and hierarchical.

- **Hierarchical Cross References** - When two signals in different blocks are connected to each other, CRefer creates hierarchical cross reference for them.
- **Flat Cross References** - When two signals in the same block are connected to each other, CRefer creates flat cross reference for them.

  Note: If you select the *Distinguish Between Ports and Offpages* check box in the Cross Referencer Options - Content Tab, CRefer distinguishes between flat and hierarchical cross references. It places flat cross references on ports or offpage symbols that do not have the HDL_PORT property on the pin. CRefer places hierarchical cross references on ports or offpage symbols that have the HDL_PORT property on a pin

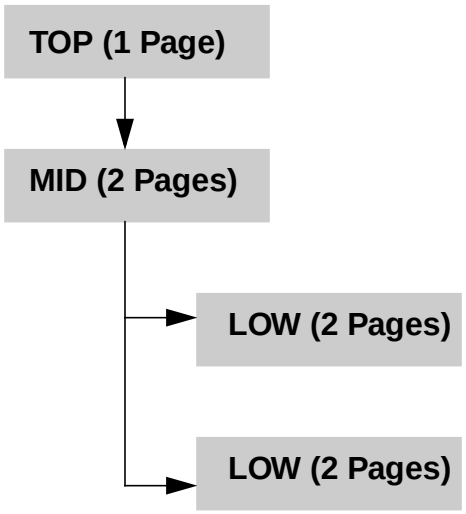
  Note: For more information about assigning properties to ports and offpage symbols, see Adding Ports or Offpage Symbols to Signals on page 468.

How the Nature of a Design Influences Cross Referencing

A design is either flat or hierarchical. The primary difference between a flat design and a hierarchical design is that all the pages in a flat design are sequentially placed and represent a single-continued structure. A flat design is like a tall building where each floor represents a single-continued structure. You cannot reach to the fourth floor from the second floor without passing through the third floor. Similarly, to process page 4 in a flat schematic you need to understand the contents of page 3. There is no bypass or alternate path.

A hierarchical design, on the other hand, is different. It is like a tree, which allows for multiple branches. A tree has multiple levels of hierarchy. The stem may divide into two main branches. Each main branch may have multiple sub-branches. Each sub-branch may have either more sub-branches or it may end on a leaf node. Like a tree, a hierarchical design has multiple levels of hierarchy. Each level of hierarchy consists of a block, which can have a

single page or may have multiple pages. Within the same block, the design behaves like a flat design. The following figure represents a hierarchical design.



In the above design, `TOP` represents the parent block, which is instantiating another block named `MID`. `TOP` and `MID` have a parent-child relationship. `TOP` represents the highest level (level 0). `MID` represents the next level (level 1). `MID` is parent to two instances of the block named LOW. Both instances of the block `LOW` are at the same level of hierarchy (level 2). Notice that the design has 3 blocks: `TOP`, `MID`, and `LOW`, each of which has a different number of pages.

If you have both the signals that are being cross-referenced in the same block, then the cross references generated are flat. Flat cross references are assigned on the signals that belong to the same block. For more information about the format of flat cross references, see Flat Cross References on page 17-451.

If you generate cross references across multiple blocks, then the cross references generated are hierarchical, and these cross references have the format described in the section on Hierarchical Cross References on page 17-452.

Flat Cross References

Flat cross references are marked in the schematic on those ports or offpage symbols that link to signals on the same block. The format of a flat cross references is:

```
Sheet# Ygrid Xgrid [Type]
```

Where:

Sheet#	Represents the drawing sheet number.
Ygrid and Xgrid	Represents the grid coordinates where the signal appears. The grid coordinates are determined using the page border grid. If you select the *Omit Zone Information* check box in the Cross Referencer Options - Content Tab, then the grid coordinates are not displayed.
Type	Represents the signal's I/O type. The I/O type is blank if it cannot be determined or if you have selected the *Omit Input/ Output Arrows* check box in the Cross Referencer Options - Format Tab. For more information about signal types, see I/O Types on page 17-486.

Example of a Flat Cross Reference

A signal on page 1 that is input on page 3 would receive a cross reference label similar to:

```
3A4<
```

This cross reference specifies that the signal appears as input at the zone A4 on page 3.

Hierarchical Cross References

You can direct CRefer to generate hierarchical cross references either on the original schematic view (*sch_1*) or in a new view (*schcref_1)*. By default, hierarchical cross references are generated on the original schematic view. You can select the *Generate Flattened Schematic* check box in the Cross Referencer Options - Content Tab, to specify that cross references are generated in the *schcref_1* view, which is created under the top-level cell of the current project. The design is flattened, copied to the *schcref_1* view, and cross references are put in it.

The format of a hierarchical cross reference is:

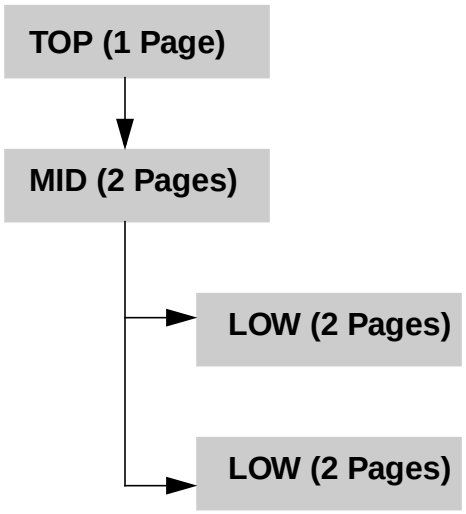
```
[Signal_Name@][_][Block_Name_@] Sheet# Ygrid Xgrid [Type]
```

Where:

Signal_Name@	If the signal is going up in the hierarchy, `Signal_Name` represents the name of the signal connected to the pin in the parent block. If the signal is going down in the hierarchy, `Signal_Name` represents the name of the signal in the child block to which it is connected. **Note:** The `Signal_Name@` is displayed if the *Show Signal Names in Hierarchical Cross References* check box is selected in Cross Referencer Options - Content tab.
Block_Name_@	Denotes that the signal comes from the specified block. For the parent block, the cross reference will include the name of the child block. However for the child block, the block name will be the name of the child itself. **Note:** The `Block_Name@` is displayed if the *Show Block Names in Hierarchical Cross References* check box is selected in Cross Referencer Options - Content tab.
Sheet#	Denotes the drawing sheet number. See Design Sheet Variable Support on page 17-491 for details about assigning design sheet variables on your design.
YGrid and Xgrid	Represents the grid coordinates where the signal appears. The grid coordinates are determined using the page border grid.
Type	Represents whether the signal is going up or down in the hierarchy. The type is blank if you have selected the *Omit Hierarchical Arrows* check box in the Cross Referencer Options - Format Tab.

Example of a Hierarchical Cross Reference

The following figure represents a hierarchical design.



In the hierarchical design displayed above, if the signal CLK in TOP is connected to the pin A in MID, then the cross reference of the pin A in MID will be:

```
1C7^
```

This cross reference indicates that the signal CLK connected to the pin A in MID is coming from the zone C7 (Ygrid=C, Xgrid=7) of page 1 of TOP. The ^ character indicates that CLK is a hierarchical signal.

However, if you have selected the *Show Signal Names in Hierarchical Cross References* check box and the *Show Block Names in Hierarchical Cross References* check box, then the cross reference for the signal CLK would be like:

```
CLK@_TOP_@1C7^
```

CRefer will also append a cross reference to the signal CLK in TOP. This cross reference will be:

```
2B8v
```

This cross reference indicates the signal CLK is connected to the pin corresponding to the zone B8 on page 2.

Note: If the signal name used in a hierarchical block does not match with the pin name used in the symbol for that block then CRefer puts a cross reference with the following syntax:

*<signal_name_at_top_level>-<Page_number_at_top_level>[path_name_of_block]^*

For example, the following cross reference is placed when the signal name used in a hierarchical block does not exactly match with the pin name used in the symbol for that block:

```
B<0>-1[I1]^
```

Note: The *schcref_1* view is an output only view created by CRefer. Do not make any design changes to it as CRefer cross references the design based on the *sch_1* view. Each time you cross-reference the design, the *schcref_1* view is created again. Therefore, any changes you make in the *schcref_1* view would be lost.

How CRefer Cross-References a Design

Cross-referencing a Design

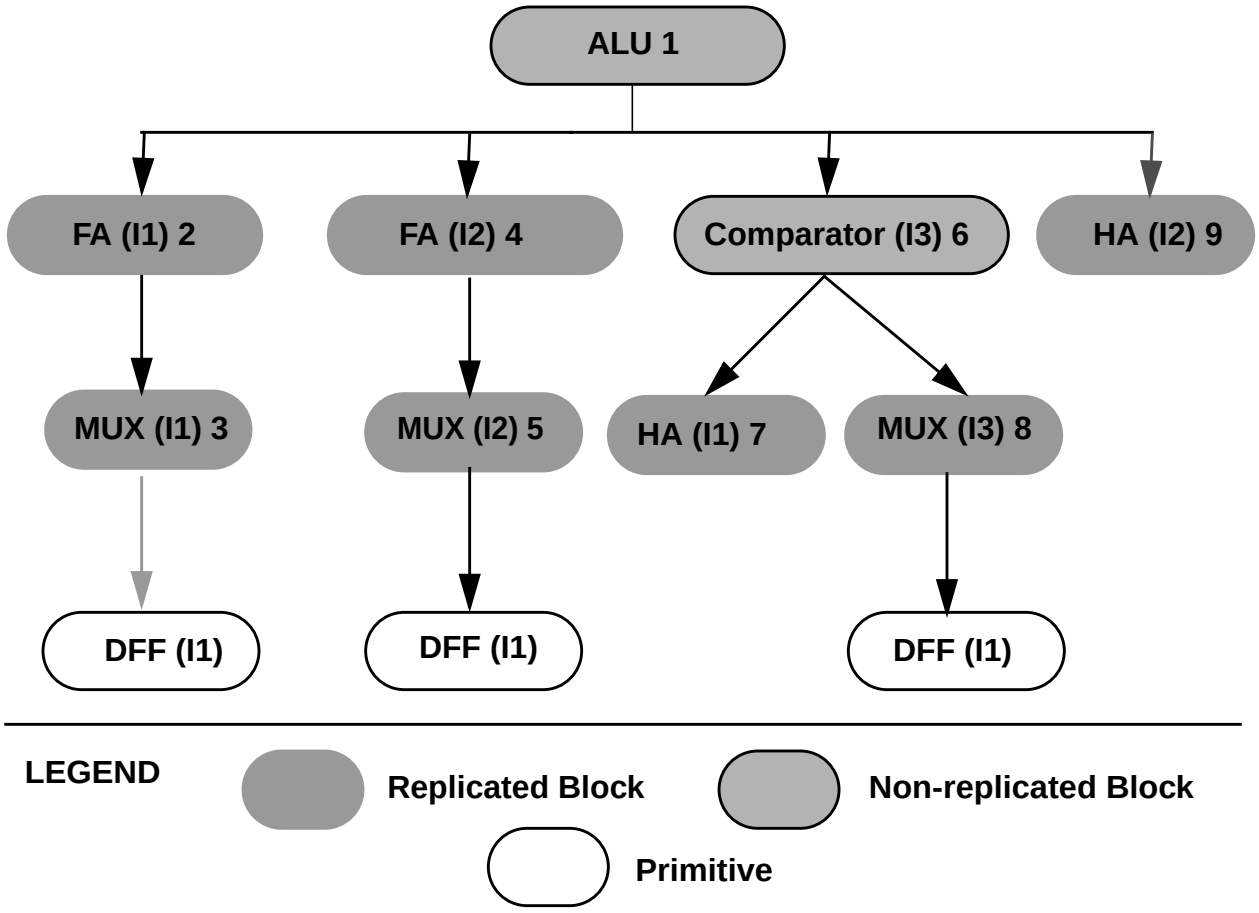
When you cross-reference a design, CRefer first parses the design and differentiates blocks on the following 2 count:

- Replicated versus non-replicated—CRefer treats a block as replicated if it is a hierarchical block and if it is being used at two or more places in the design. If a particular block is being replicated, then the entire subtree under it; that is, all blocks contained within it are also treated as replicated.
- Read-only versus writable—Based on whether a block is read-only or writable, CRefer places the cross references on the schematic or in the OPF file (`cref.opf`). Concept HDL reads the `cref.opf` file and displays the cross references on the schematic.

Depending upon your design, you may have multiple scenarios. For example, consider the design in the ALU Hierarchical Design figure on page 456. Notice that the design is named `ALU`, and this design includes multiple levels of hierarchical blocks. The `ALU` block contains four blocks, which includes two instances of `FA`, one instance of `Comparator`, and one instance of `HA`. The `FA` block is being replicated. CRefer will treat the entire hierarchy under both the `FA` blocks as replicated.

Notice that the `MUX` block is also being replicated. There is one instance of `MUX` under the `FA (I1)` block and another instance under the `Comparator` block. CRefer will treat both instances of `MUX` as replicated hierarchy. This means that the `MUX (I1)` block and its child block `DEF (I1)` are both treated as replicated hierarchy.

Figure 17-1 ALU Hierarchical Design



Naming Convention in Block: Block name (instance name) Design current sheet value

Notice that the HA block is also being replicated. One HA block exists below the `Comparator` block, while another instance of the HA block exists below the ALU block.

In summary, CRefer will treat the ALU and `Comparator` blocks as non-replicated hierarchy and all other blocks as replicated hierarchy.

Cross References for Non-replicated Hierarchy

Note: For non-replicated blocks, that is ALU and `Comparator`, CRefer annotates the properties on the placeholders specified on ports or offpage symbols. If you have not defined

any placeholders on the ports or offpage symbols on the schematic, then CRefer creates placeholders and attaches cross references to the ports or offpage symbols.

Note: Create placeholders on the ports or offpage symbols to ensure that cross annotations are placed on the schematic. The placeholder should be a property name - value pair.

Note: If there are any XR or $XR properties existing on the schematic that are not attached to any ports or offpage symbols, then CRefer deletes those properties.

Note: If you have not used ports or offpage symbols on the signals that need be cross-referenced, then you need to select the *Cref Signals Not Connected to Flag Bodies* check box in the Cross Referencer Options - Content tab to allow CRefer to cross-reference these signals.

Cross References for Replicated Hierarchy

For all replicated hierarchy, such as FA (see ALU Hierarchical Design on page 17-456), CRefer creates cross references in the `cref.opf` file in the `sch_1` view of the top-level cell (ALU). The `cref.opf` file is a binary file.

It is required that you add ports or offpage symbols on the schematic on all the nets that you want to be cross-referenced.

When CRefer runs out of placeholders, it dumps all the remaining cross references on the top of last place holder. The following message is also generated:

```
Signal <signal_name> at <location_of_SIG_NAME> required
<number_of_required_placeholders> placeholders on
<name_of_the_attached_plumbing_body>.
```

If there is no placeholder attached to a port or offpage symbol, CRefer creates placeholders and annotates cross references.

Cross References for Read Only Blocks

If you have read-only blocks in the schematic, then CRefer will add cross references in the `cref.opf` file in the `sch_1` view of the top-level cell (ALU).

Viewing Cross References in Concept HDL

On running Concept HDL, select the Occurrence edit mode. The Occurrence edit mode is used by Concept HDL to read occurrence-specific properties corresponding to a replicated

hierarchy. In a replicated hierarchy, the same block in a design is used in multiple places and can be assigned different set of properties in different places. You may have different properties for drawings, components, nets, and pins.

CRefer in PSD 14.2 allows you to have occurrence-specific cross annotations. The information about these cross annotations is available in the `cref.opf` file. In the Occurrence edit mode, Concept HDL detects the placeholders on the ports or offpage symbols and if there are not any placeholder, Concept HDL creates placeholders. For each placeholders, Concept HDL replaces the property value by reading the corresponding property in the `cref.opf` file. For example, if the value of the OPF property `$XR0` in the `cref.opf` file is `5B3<`, Concept HDL will display the property `5B3<` on the placeholder created by CRefer, where `B3` represents the zone coordinates and < signifies that the signal is an input signal.

Concept HDL also displays `$XR` properties attached to the ports or offpage symbols in the *Attribute* form in the Occurrence edit mode.

Note: Ensure that the property `OFFPAGE=TRUE` is attached to all ports and offpage symbols. If you have not assigned the `OFFPAGE=TRUE` property to a port, then you will not see the cross reference for that port, although the property information for the same is available in the `cref.opf` file.

Controlling CRefer Annotations Using UI Options

1. **Cref Signals Not Connected To Flagbodies -** By default, CRefer ignores the signals that do not have flag or port bodies attached. However, if you select the *Cref Signals Not Connected To Flagbodies* check box, then CRefer will attach the cross references to the `SIG_NAME` property of signals.

   Note: If you select the *Cref Signals Not Connected To Flagbodies* check box, then CRefer will not be able to correctly cross-reference the signals in replicated hierarchy and read-only blocks in the `sch_1` view.

2. **Redo Placement of Crefs -** By default, CRefer retains the previous placement of cross references. If you select the *Redo Placement of Crefs* check box, then the previous placement is lost, and CRefer uses the original placeholders on the ports or offpage symbols to place cross references.

3. **Add Crefs as Hard Properties -** All placeholders created by CRefer are soft properties (`$XR`). However, if you select the *Add Crefs as Hard Properties* check box in the Cross Referencer Options - Format tab, then all placeholders are converted into hard properties (`XR`).

Summarizing In-place Cross Referencing and Place Holder Support

CRefer uses the algorithm as displayed in the In-place Cross Referencing of Signals figure on page 460 and In-place Cross Referencing for Blocks figure on page 461 to cross-reference any design. Notice that the action performed by CRefer is based on the following factors:

- Does a signal has offpage symbols or ports?
- Does a signal has placeholders
- What is the nature of the block (replicated, non-replicated, or read-only)?

The behavior of CRefer for various types of design will be as follows:

Legacy Designs with XRs attached to SIG_NAMEs and no placeholders in library.

CRefer searches all XR properties attached to signals or ports and retains their value. However, CRefer attaches XR properties to the ports or offpage symbols instead of SIG_NAMEs.

Legacy designs with XRs attached to SIG_NAMEs. The library has been updated with placeholders, and the hier_write operation has been performed in Concept HDL.

By default, CRefer honors existing placement, but attaches the XR properties to offpage symbols. If the *Redo Placement of Crefs* check box is selected, CRefer treats the design as a new design and puts XRs in the placeholders provided in the library.

Legacy designs with a few new pages added. The library in the design has placeholders.

By default, CRefer honors existing placement and uses placeholders wherever XRs were not originally present. In case a signal had say 2 XRs initially, and is assigned 3 XRs in the next run due to changes in the design, CRefer places the new XR using its own algorithm. It will attach the cross references to ports.

New designs with placeholders.

CRefer will always use placeholders for placement and will honor any changes made in the placement in the later runs.

Figure 17-2 In-place Cross Referencing of Signals

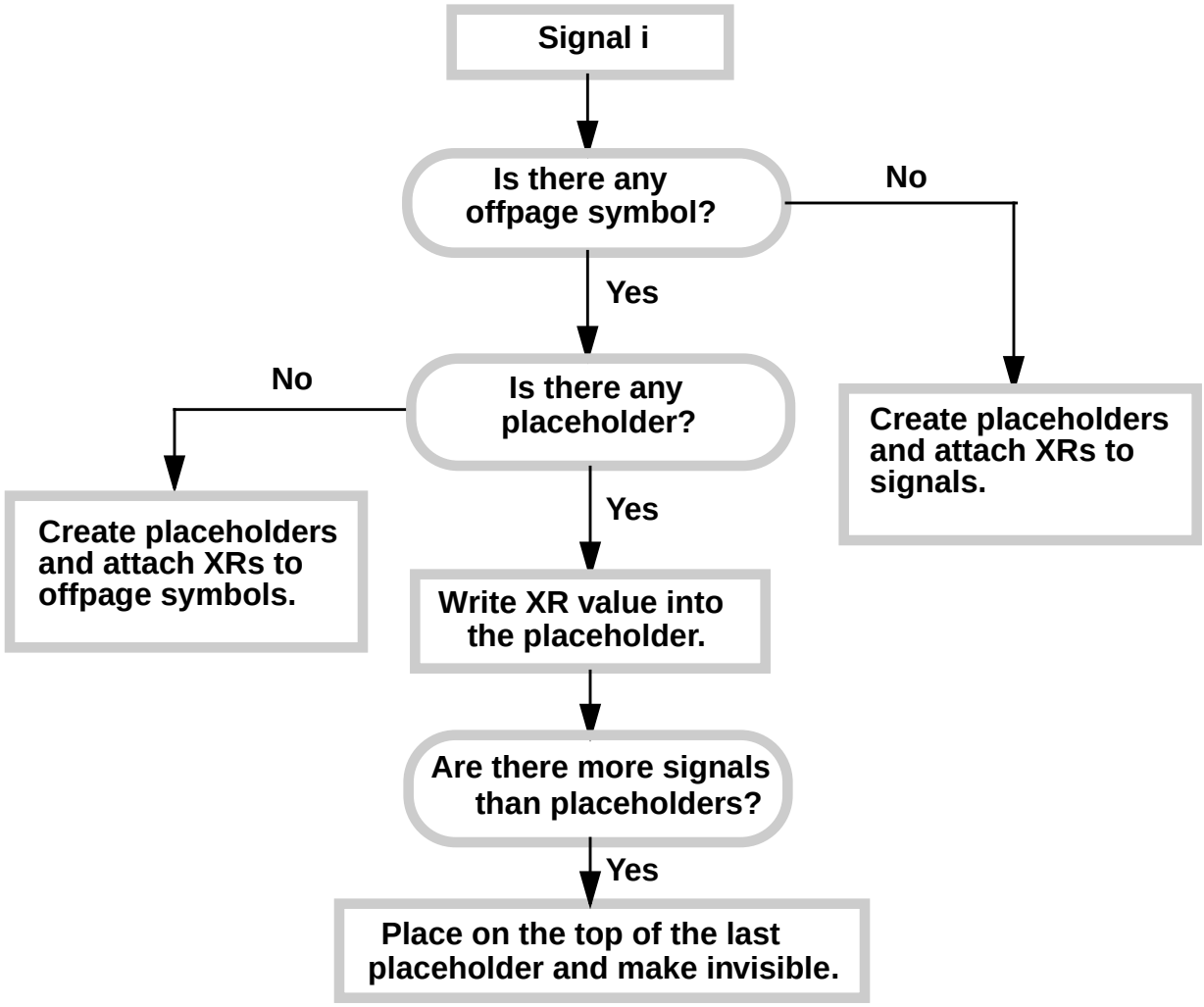
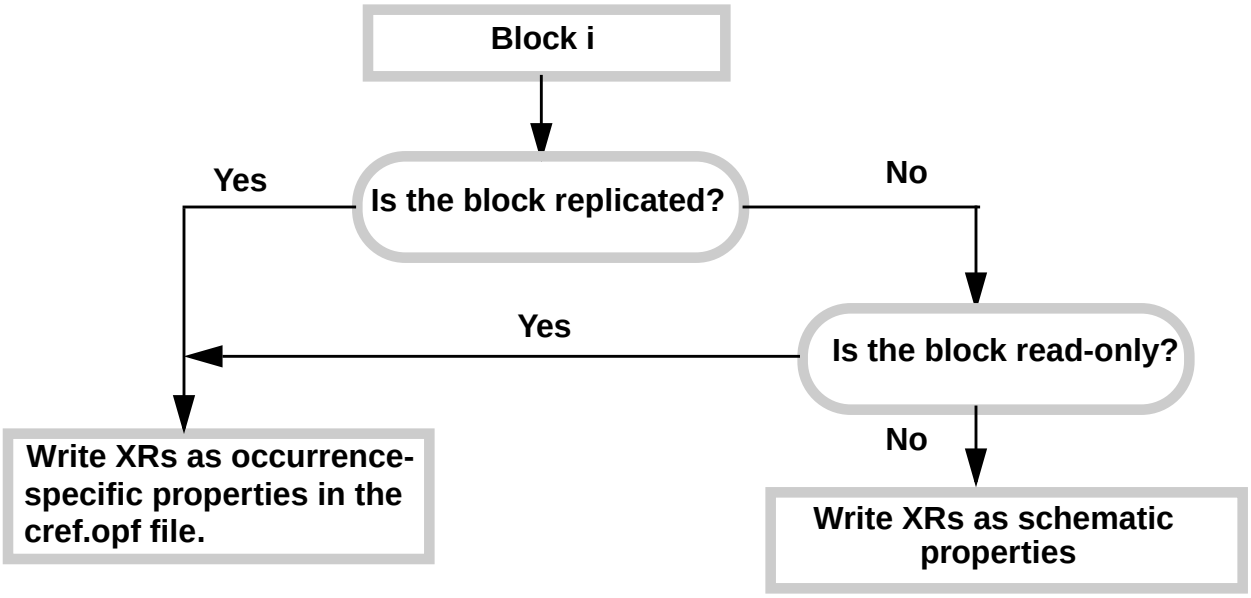

Signal i
Is there any offpage symbol?
No
Yes
Is there any placeholder?
No
Yes
Create placeholders and attach XRs to signals.
Create placeholders and attach XRs to offpage symbols.
Write XR value into the placeholder.
Are there more signals than placeholders?
Yes
Place on the top of the last placeholder and make invisible.

Figure 17-3 In-place Cross Referencing for Blocks



Getting Started with CRefer

Before you cross-reference a design, you would be required to do one or all of the following procedures:

- Preparing the Design for Cross Referencing on page 461
- Determining the Right Cross Referencing Options on page 463
- Creating the Cref Data File for Page Borders on page 464
- Creating Custom Offpage I/O Flag Bodies on page 465
- Making Cross References Permanently Visible on page 467
- Adding Ports or Offpage Symbols to Signals on page 468

Preparing the Design for Cross Referencing

To prepare the design for cross referencing ensure that you have performed the following steps:

1. The quality of cross references generated by CRefer is directly proportional to the quality of design entered in Concept HDL schematic. You should ensure that the schematic has

enough space for CRefer to place cross references. Therefore, design the schematic with cross referencing in mind. For all signals that you want to be cross-referenced, leave enough space.

In particular, ensure that you leave enough room around flag/port/offpage symbols so that CRefer annotations are placed properly. The wire-to-wire spacing between nets where flag/port/offpage symbols are attached should be increased as necessary.

2. It is recommended that you add a $XR<n> placeholder on the port or offpage symbol that you want to be cross-referenced. This will ensure that you get cross references at the desired place on the schematic. For more information about adding placeholders, see Adding Placeholders on Ports or Offpage Symbols on page 495.

3. Ensure that the property OFFPAGE=TRUE is attached to all ports and offpage symbols.

4. CRefer will know whether or not a signal needs to be cross-referenced only if it has an offpage body, port, or offpage symbol attached to it. Therefore, ensure that you have attached an offpage body, port, or offpage symbol to every signal that you want to be cross-referenced. You may, however, direct CRefer to cross-reference all signals whether or not they are connected to flag bodies or ports. For this, use the *Cref Signals Not Connected to Flag Bodies* check box in the Cross Referencer Options - Content Tab.

5. Ensure that all the schematic pages in your design use one of the page borders available in the standard library. You may also use custom page borders in your design.

6. You can add custom text for CRefer variables to have more explicit annotations. For example, you can place custom text such as “This block goes to page <*CREF_TO_LIST*>” in the schematic. To add custom text, use the Concept Options and Custom Text dialog boxes in Concept HDL.

7. If you want to change the order of the modules (that is hierarchical blocks), use the Module Ordering dialog box in Concept HDL. Similarly, if you want to exclude certain modules from the hierarchical design, then exclude those modules in the *Module Ordering* dialog box.

8. Ensure that you have packaged the design before cross referencing it. This is important if you want to generate the Parts Cross Reference report (crefparts.txt). For more information about packaging a design, see the Cadence document *Packager-XL Reference*.

9. If you want to see OPF properties as visible in the *schcref_1* view, then backannotate the design.

Determining the Right Cross Referencing Options

After you have prepared the design for cross referencing, you must set the right cross referencing options for your design. The enclosed list details different scenarios and mentions the options that you must set for proper cross referencing of your design:

1. If your drawings use page borders other than those provided in the standard library, you must create the cref data file (`cref.dat`), which defines your custom page borders. Specify the path to the cref data file in the Cross Referencer Options – Cref Data File tab. This path will be written into the project file, and therefore would be automatically available the next time you cross-reference your design.
2. Ensure that you have added a description about all offpage flag bodies and ports in the cref data file for the project (even if the same description also exists in the site/Cadence level Cref data file). For more information about adding custom offpage I/O flag bodies, see Creating Custom Offpage I/O Flag Bodies on page 17-465.
3. If you want to suppress cross referencing of specific signals or generate cross references for power signals, you should list them in the cref data file. For more information about suppressing cross referencing of certain signals, see Suppressing Cross Referencing of Signals on page 17-465 or Creating Cross References for Power Signals on page 17-465.
4. If you have cross-referenced a design once and want to retain the placement of cross references during repeat cross referencing, then do not make changes to the format of cross references. For example, you should not change the display of zone, signal, or block information, or change the text size or spacing. Any change to the format of cross references will not allow for the proper placement of cross references. In such cases when you do make changes to the format of cross references, select the *Redo Placement of Crefs* check box in the Cross Referencer Options - Format Tab to ensure that the cross references are placed again.
5. If you need to split the bus at the lower level in the hierarchy, then it is recommended that you use the complete bus for the signal name at the lower level and tap its individual bits. This will provide better cross reference reports.
6. If you plan to write the cross referencing options in the project file, then use the CRefer dialog box to make the changes. Any changes made using the CRefer dialog box are stored in the project file. However, if you make any changes in cross referencing options using the command-line prompt, then those changes will be used for that CRefer run only. These changes will not be stored in the project file.
7. If you want to find the sheet number of current page or find the total number of pages present in the schematic, use sheet numbering.For more information about adding any CRefer custom variable (which includes sheet numbering variables), see Adding CRefer Custom Variables on page 17-488.

8. If you want to distinguish between hierarchical and flat cross references, select the *Distinguish Between Ports and Offpages* check box in Cross Referencer Options - Content tab. This will ensure that hierarchical cross references are placed on ports and flat cross references are placed on offpage symbols.
9. If you have multiple users who use the same cross referencing option, you can create a site project file and save the default cross referencing options in it. To create a `site.cpm` file, either use an existing projects project file, or create a dummy project and use its project file to define your site settings. For more information about creating a site project file, see the Cadence document *Concept HDL User Guide*.

Note: You can cross-reference a schematic from the command-line prompt. However, it is recommended that you use the CRefer dialog box to make all cross referencing settings, and then, if required, use the command-line prompt to cross-reference the design.

To cross-reference a design from the command-line prompt, use the following syntax:

```
creferhdl -proj <project_file> [-d]
```

where, the `-proj <project_file>` option specifies the path to the project file you want to cross-reference. Use the `–d` option to delete all the existing cross references in the design.

Creating the Cref Data File for Page Borders

To create cross references, you must use a page border for each page of the design. The Cadence *standard* library provides six standard page borders—`A SIZE PAGE` to `F SIZE PAGE`. These page borders are the first versions and can be identified by their version numbers (1 in these cases). The details of these page borders are available in the cref.dat file.

You can use the same or different page borders for the pages of the design. To use different page borders for various pages of the design, you must define the page borders in the `cref.dat` file in the sequential order.

You can:

1. Use a standard page border.
2. Modify a standard page border and use it.
3. Create and use your own page border.

Creating Custom Offpage I/O Flag Bodies

You can create and use your own offpage bodies. The body shape does not matter, but the property COMMENT_BODY=TRUE must be attached to the body drawing. Version 1 of the OFFPAGE flag body is shown below as an example.

To identify your custom offpage bodies; you must specify their I/O type in the cref.dat file. Each version of the body must be declared separately. Typically, you need six versions of an I/O flag: input, output, and bi-directional flags facing both left and right.

The syntax for declaring offpage bodies is:

```
INFLAG “flag name ” VERSION number
OUTFLAG “flag name ” VERSION number
BIFLAG “flag name ” VERSION number
```

For example, you may declare six versions of an offpage body named CROSSFLAG as follows:

```
INFLAG “CROSSFLAG” VERSION 1
OUTFLAG “CROSSFLAG” VERSION 2
BIFLAG “CROSSFLAG” VERSION 3
INFLAG “CROSSFLAG” VERSION 4
OUTFLAG “CROSSFLAG” VERSION 5
BIFLAG “CROSSFLAG” VERSION 6
```

Creating Cross References for Power Signals

By default, the power signals-VCC, NC, VSS, GND, 0, and 1-are not cross-referenced. To create cross references for the power signals, define the NONTRIVIALNET directive in the cref.dat file:

```
NONTRIVIALNET “Signal name in quotes”
```

Suppressing Cross Referencing of Signals

You can suppress cross referencing of any signal by specifying the signal name in the cref.dat file. By default, the power signals-VCC, NC, VSS, GND, 0, and 1-are not cross-referenced. To suppress cross referencing of signals other than the above power signals, use the TRIVIALNET directive in the *cref.dat* file:

```
TRIVIALNET “Signal name in quotes”
```

Note: You may often find the `TRIVIALNET` directive useful to ignore global signals while cross referencing a design.

Determining Coordinates

To determine the xmark and ymark coordinates in a page:

1. Open the schematic in Concept HDL.
2. Choose *Display > Coordinate*.

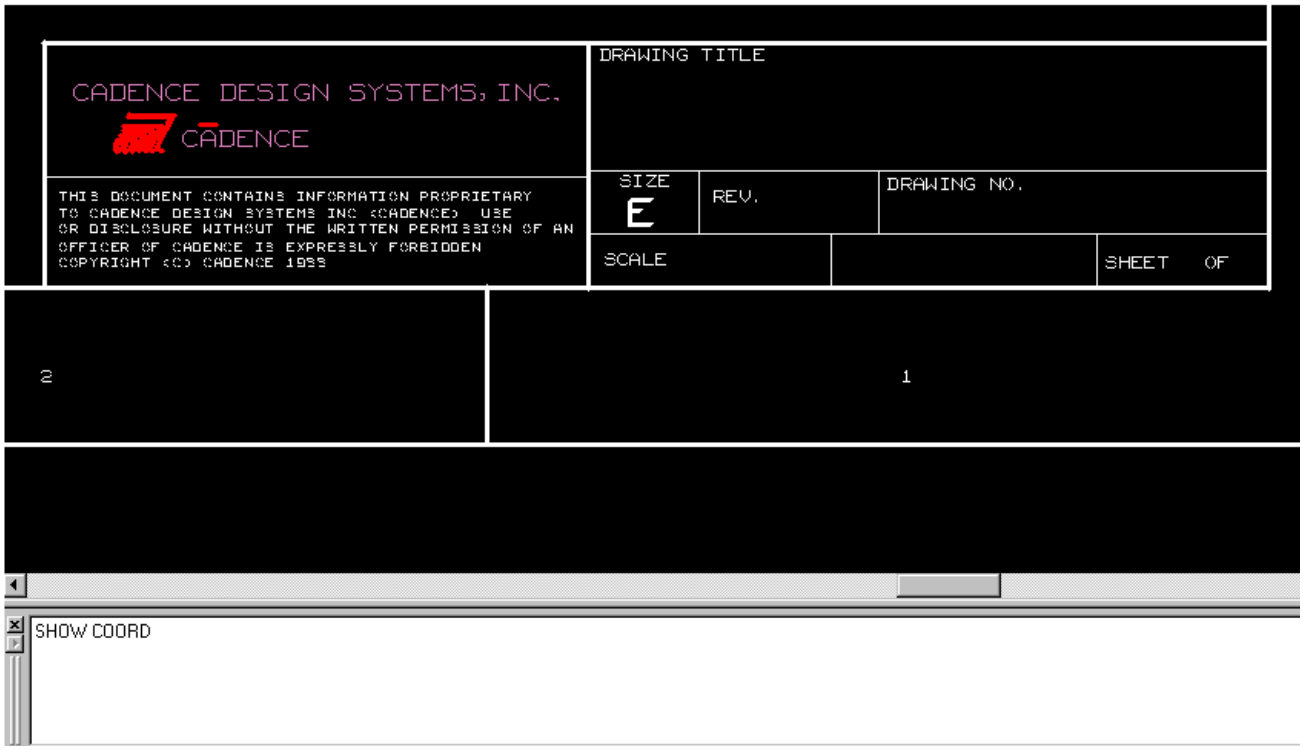
   Note: You can also use the Console Window to display coordinates. For this, type `SHOW COORDINATE` (or SHOW COORD) in the Concept HDL console and press Enter.
3. Click on the number or letter of the Zone (For example, 1 or A). The coordinate value is displayed in the Concept HDL status bar.

   Note: Click the mouse directly on the top of the number or letter; do not click on the boundary lines.
4. If the Zone number or Zone letter is to the left of the Origin, subtract the X coordinate from the Origin X coordinate.
5. If the Zone number or Zone letter is to the right of the Origin, add the X coordinate to the Origin X coordinate.

The resulting number is the value of xmark. The same calculation also applies to ymark.

To understand how to determine coordinates, consider the following example:

Part of a page border is displayed below. The origin of the page border is located at the center of the page. All zone numbers and letters displayed are to the right of the origin.



The `Show COORD` command is entered. You can click on any Zone letter or Zone number in the graphic to determine its coordinate. To find the value of the coordinates for the Zone number “1”, click on the number 1 in the above graphic. Concept HDL will return the xmark and ymark for the Zone number “1”.

Making Cross References Permanently Visible

To make cross references permanently visible:

1. Start Concept HDL and edit the schematic.
2. Group the invisible $XR properties.

   ```
   FIND $XR*
   ```

3. Make the property values visible.

   ```
   DISPLAY VALUE “A”
   ```

   ‘A’ is the group name assigned by the *Find* command. If you have already created some groups in the current Concept HDL session, the group you created with the *Find* command may have another name.

4. Use the *Next* command in Concept HDL to automatically zoom in on each property.

Adding Ports or Offpage Symbols to Signals

Adding offpage symbols

Add offpage symbols to all signals that will have flat (offpage/onpage) cross references.

➤ To add an offpage symbol, assign the following property set for the SRC, ONSRC, ONDST, and DST symbols in the existing libraries:

```
OFFPAGE=SRC|ONSRC|ONDST|DST
```

Note: Ensure that the above property is added on the original symbols and not on the pins.

Note: The current implementation of CRefer does not distinguish the difference in value that you assign to the OFFPAGE property. These values are useful for future enhancements.

Adding Ports

Add ports to all signals that will have hierarchical cross references. To add a port:

1. Assign the property OFFPAGE=TRUE to the port.
2. Assign the following property set to the port:

```
HDL_PORT=IN|OUT|INOUT
```

Using CRefer

You can directly cross-reference a design or you can first customize the cross referencing options and then cross-reference the design. To cross-reference a design properly, you would be required to perform one or all of the following options:

- Cross-Referencing the Design on page 469
- Changing the Cref Data File on page 471
- Configuring Run and Write Options on page 472
- Configuring Formatting Options on page 474
- Defining Output Reports on page 476

■ Deleting Cross References on page 478

Cross-Referencing the Design

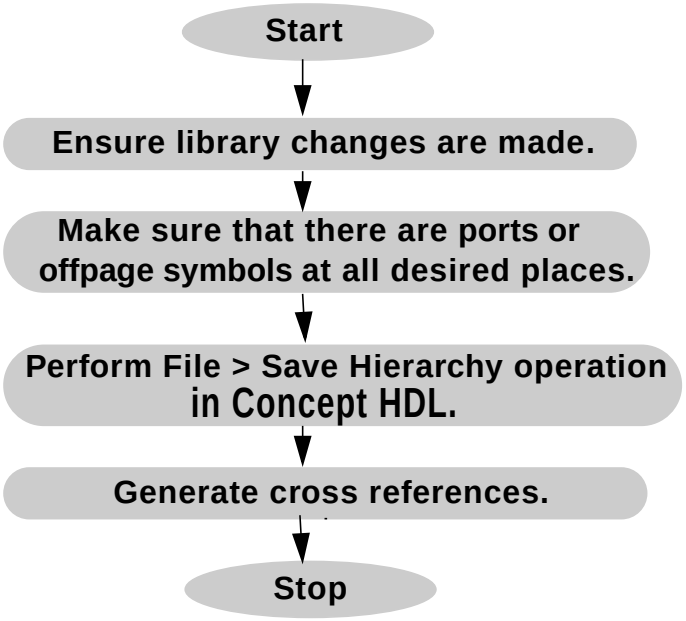
To cross-reference a design, you need to perform the following steps:

1. Ensure that the librarian has made the necessary changes as mentioned in Placeholder Support on page 495.
2. Make sure that you have added ports or offpage symbols to all signals that require cross annotation. This will ensure that cross-references assigned by CRefer will move along with ports allowing for predictable placement in the schematic.
3. Open the schematic and perform the `hier_write` operation (*File > Save Hierarchy*) on the schematic of the root drawing. This operation will assign the `PATH` property to all the offpage symbols.
4. Close the schematic by selecting *File > Exit* in Concept HDL.
5. cross-reference the design.

After the design is cross-referenced, you may open Concept HDL and view cross references. You will see that CRefer has made annotations for the entire design. CRefer has also substituted the variable values for all custom text that includes CRefer-specific custom variables.

The flow-chart in the Cross Referencing Flow figure on page 470 summarizes the main cross-referencing steps:

Figure 17-4 Cross Referencing Flow


Start
Ensure library changes are made.
Make sure that there are ports or offpage symbols at all desired places.
Perform File > Save Hierarchy operation in Concept HDL.
Generate cross references.
Stop

Generating Cross References for a Design

You can cross-reference a design using the CRefer dialog box (that is by using the user interface) or by running CRefer from the command-line prompt.

You can invoke the CRefer dialog box in one of the following two ways:

1. Select *Tools > CRefer* from Project Manager menu bar.

   CRefer dialog box displays with the name of the project file selected in the Project Manager as seeded in the *Project File* field.

2. Double-click on the creferui.exe file in Windows Explorer.

   CRefer displays the file browser and you are asked to select the project file that is to be cross-referenced.

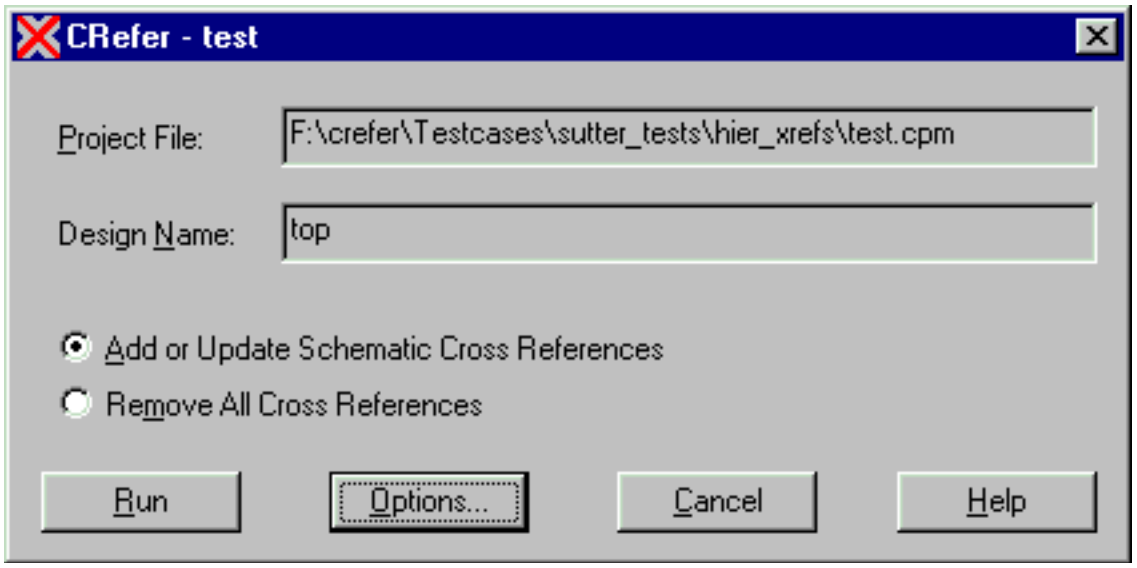
3. To cross-reference a design from the command-line prompt, use the following syntax:

   ```
   creferhdl -proj <project_file>
   ```

   where, the `-proj <project_file>` option specifies the path to the project file you want to cross-reference. Use the `-d` option to delete all the existing cross references in the design.

Note: Before you cross-reference a design from the command-line prompt, set all cross referencing options using the Cross Referencer Options Dialog Box.

CRefer Dialog Box



After you have invoked the CRefer dialog box, use the following procedure to cross-reference the design.

1. If you want to change the Cross Referencer settings, click on the *Options* button.

   The Cross Referencer Options dialog box is displayed. You can change the Cross Referencer settings here. When you have changed the settings, click on the *OK* button to return to the CRefer dialog box.

2. To cross-reference a design, select the *Add or Update Cross References* radio button.

3. To start the cross referencing process, click on the *Run* button.

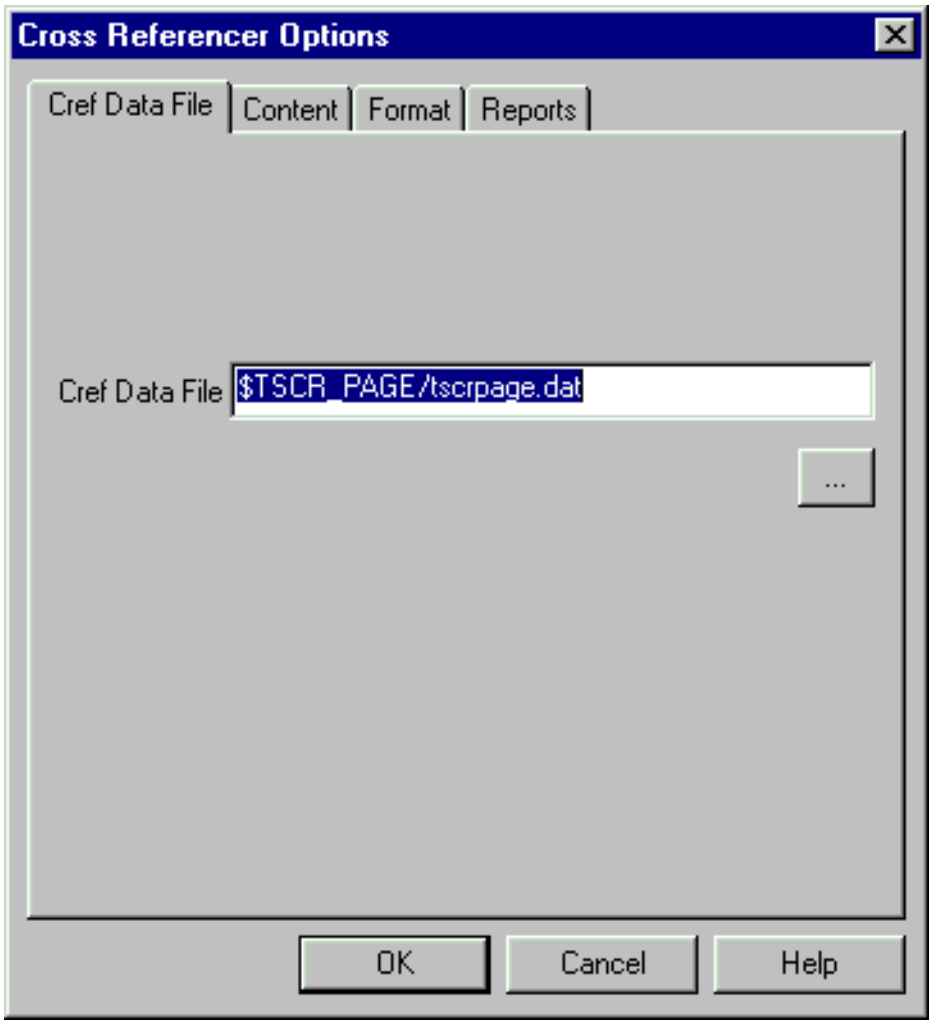
   The CRefer Progress Window is displayed. The design is being cross-referenced.

Changing the Cref Data File

By default, CRefer uses the CSF search to locate the Cref data file specified in the project file to obtain the default page border, offpage bodies, and information about signals that need be cross-referenced.

You can change the Cref data file by performing the following step in the Cross Referencer Options - Cref Data File Tab.

Cross Referencer Options - Cref Data File Tab

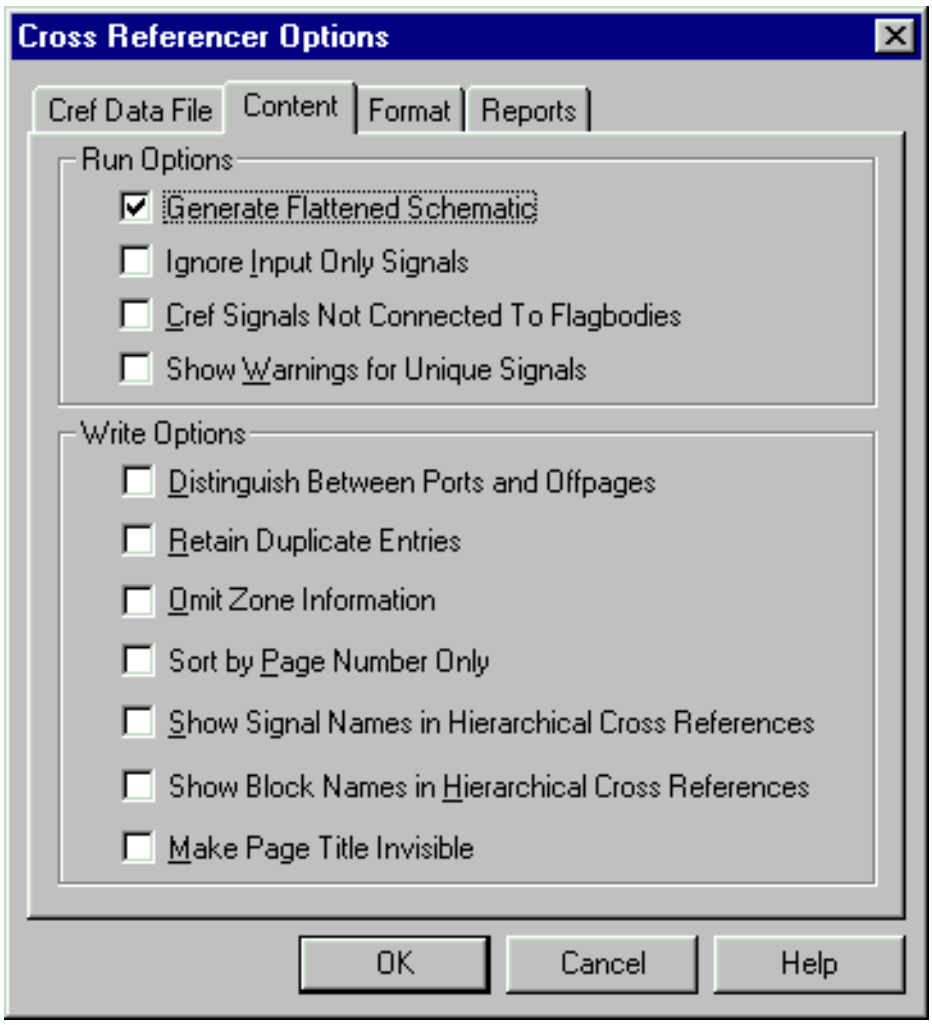


➤ Enter the path of the cref data file in the *Cref Data File* field or browse to the new file by using the *browse* button.

Configuring Run and Write Options

To configure the default run and write options, use the Cross Referencer Options - Content tab. You can select the signals that CRefer can ignore while running. You can also define how CRefer should write the cross references in the project (.cpm) file.

Cross Referencer Options - Content Tab



1. To create a new flattened view (schcref_1) view in the top-level cell for the current project) for the cross-referenced design, select the *Generate Flattened Schematic* check box.
2. To ignore the signals that are input only, select the *Ignore Input Only Signals* check box. These signals will also be ignored in the schematic reports.
3. By default, CRefer ignores the signals that do not have flag bodies or ports. If you want CRefer to read and process signals that do not have flag bodies or ports attached, select the *Cref Signals Not Connected to Flag Bodies* check box. These signals will also be placed in the schematic reports.
4. To display the warnings for signal names that occur only once in the design, select the *Show Warnings for Unique Signals* check box.

5. To specify that CRefer places hierarchical cross references on ports and flat cross references on offpage symbols, select the *Distinguish Between Ports and Offpages* check box.
6. To retain the duplicate entries in the signal and part cross references, select the *Retain Duplicate Entries* check box.
7. To annotate information only about page numbers and not about zones, select the *Omit Zone Information* check box.
8. To sort the signal cross references only by page number, and not by the input/output type, select the *Sort by Page Number Only* check box.
9. To display the signal names in hierarchical cross references, select the *Show Signal Names in Hierarchical Cross References* check box.
10. To write the name of the block where the signal originates in the cross-reference information for hierarchical designs, select the S*how Block Names in Hierarchical Cross References* check box.

    The cross reference will appear as:

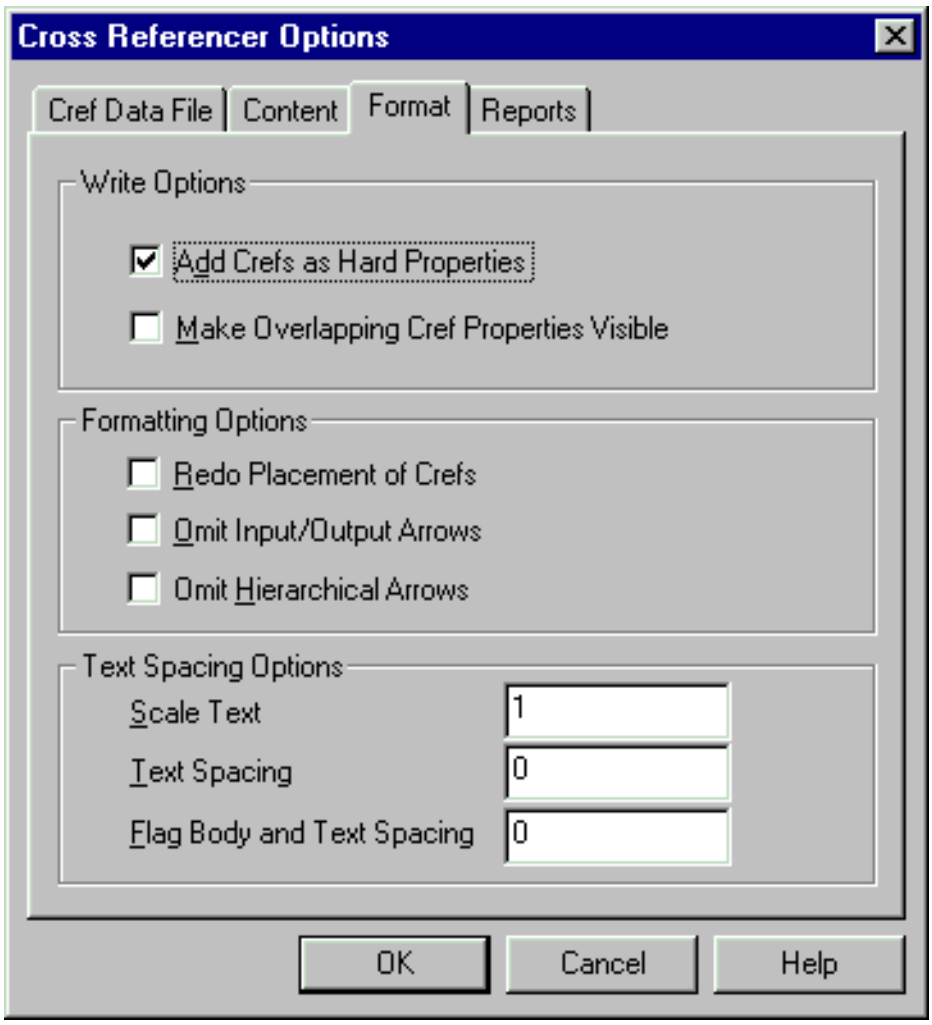
    ```
    [Signal_Name@][_BlockName_@]Page#Ygrid Xgrid[Type]
    ```

11. To make the XR page title invisible, select the *Make Page Title Invisible* check box.

Configuring Formatting Options

The Cross Referencer Options - Format tab allows you to configure CRefer formatting options. You can use this tab to specify whether or not existing crefs will be used. You can also specify to ignore input, output, and hierarchical arrows. Further, you can define the text size of annotated CRefer properties and the space between each property.

Cross Referencer Options - Format Tab



To configure formatting options, perform the following steps:

1. To add Cref properties as hard properties (XR), select the *Add Crefs as Hard Properties* check box. By default, CRefer add cross references as soft properties ($XR).
2. To make the properties visible (that cannot be displayed on the schematic due to lack of space) on the schematic, select the *Make Overlapping Properties Visible* check box.
3. To override the previous cross-reference placement in a design (this information is available if the design has been cross-referenced in the past.), select the *Redo Placement of Crefs* check box.

   By default, CRefer reuses the existing cross-reference information.

4. To omit the writing of the characters (that is, "<", ">", and "<>") for I/O types, select the *Omit Input/Output Arrows* check box.
5. To omit the writing of the characters (that is, “^" and “v") for I/O types, select the *Omit Hierarchical Arrows* check box.
6. To increase or decrease the text size of cross references, type the scale factor (the text size in relation to the default display) in the *Scale Text* field.

   Example - To scale the text size to half of the original size, enter 0.5 in the Scale Text field. To scale the text size to twice the original size, enter 2 in the Scale Text field. The setting 0.5 can be used when the schematic is densely packed.

   Note: When you specify the Scale Text, keep in mind the smallest pages in the design.
7. To increase or reduce the space between two cross references, type a number in the *Text Spacing* field.

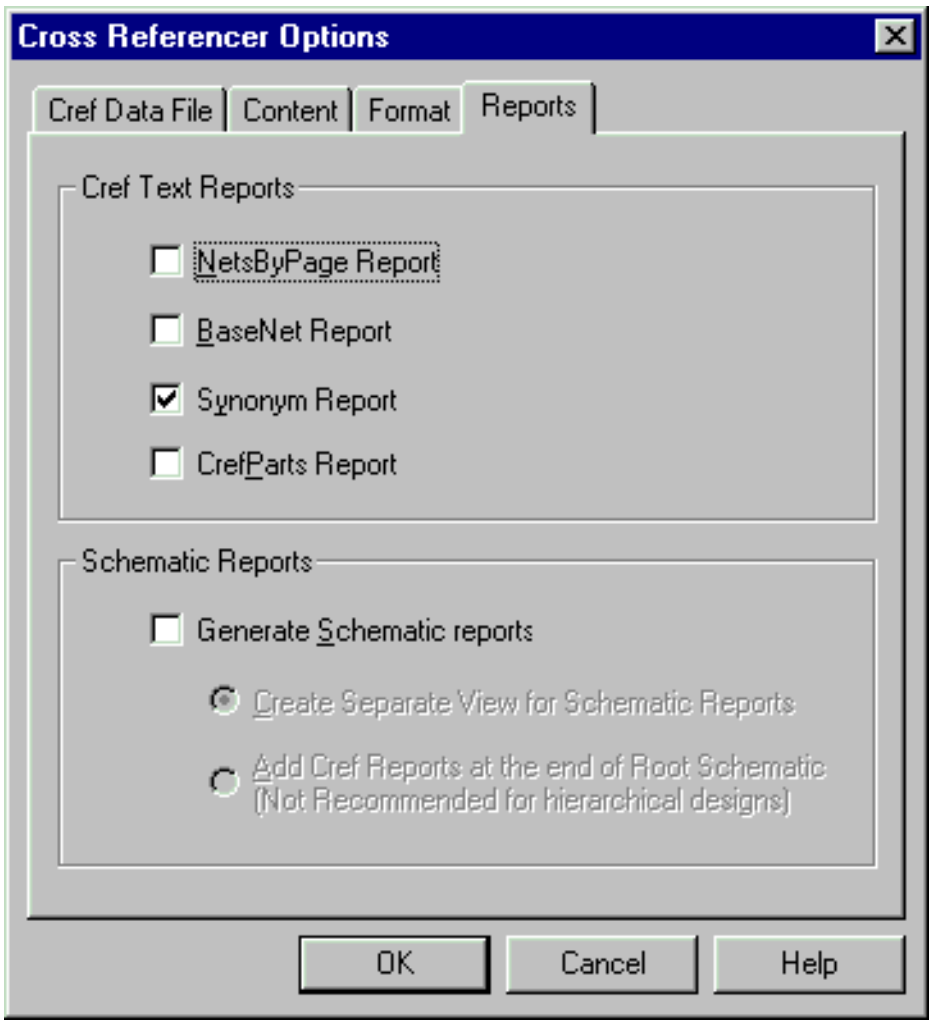
   Note: CRefer uses default Concept HDL coordinates for text spacing.
8. To increase or reduce the space between the flag body and the cross-reference text, type a number in the *Flag Body and Text Spacing* field. To reduce the spacing, use a negative value; otherwise, use a positive value.

   Note: CRefer uses default Concept HDL coordinates for flag body and text spacing.

Defining Output Reports

To define the types of reports to be generated as outputs, use the Cross Referencer Options - Reports tab. For example, you can generate Synonym reports.

Cross Referencer Options - Reports Tab



To generate different reports, perform the following steps:

1. To create a report that contains the list of nets and their base nets grouped by page, select the *NetsByPage Report* check box.
2. To create a report that contains the signal cross-reference information grouped according to the design cells, select the *BaseNet Report* check box. The Basenets report also includes the direction characters with the signal and synonym information.
3. To create a report that traces a net across a hierarchical design, select the *Synonym Report* check box.
4. To create a report that contains information about all unit cross references for the entire design, select the *CrefParts Report* check box. The Crefparts report includes

information about the path property attached to the cell, the symbol name, and the cross references.

5. To create signal part reference and part cross reference schematic reports, select the *Generate Schematic Reports* check box.

   Notice that the *Create Separate View for Schematic Reports* radio button is selected by default. CRefer creates the two schematic reports in a separate view named *crefout*, under the root design. This view contains two pages, one each for signal and part cross reference reports.

   Note: The reports added by CRefer at the end of the schematic during any previous run are automatically deleted.

6. To add signal and part cross reference reports at the end of the root schematic, select the *Add Cref Reports at the End of the Root Schematic* radio button.

   Note: It is not recommended to add signal and part cross reference reports at the end of the root schematic for hierarchical designs.

   Note: If you have selected the *Add Cref Reports at the End of the Root Schematic* radio button and later you decide to create schematic reports in a separate view, select the *Create Separate View for Schematic Reports* radio button.

Deleting Cross References

To delete the cross references from a design, use the CRefer dialog box.

1. Select the *Remove All Cross References* radio button in the CRefer dialog box.
2. To remove the cross references, click on the *Run* button.

   The CRefer Progress window displays that cross references are being deleted.

   Note: To delete the cross-reference for a design from the command-line prompt, use the following syntax:

   ```
   creferhdl -proj <project_file> -d
   ```

   where, the `-proj <project_file>` option specifies the path to the project file you want to cross-reference, and the `-d` option is used to delete all the existing cross references in the design.

How CRefer Deletes Cross References

When deleting cross references, CRefer will:

- Delete all properties whether hard or soft. Crefs for both XR and $XR properties will be deleted.
- Delete the XR_PAGE_TITLE from the page borders.
- Delete the extra pages added to the schematic by earlier cross referencing.
- Delete the values of custom variables used in custom text.
- Delete the `cref.opf` file.

When deleting cross references, CRefer will not:

- Delete reports from the *rptcref_1* view.
- Delete properties from the *schcref_1* view. Since the *schcref_1* view remains unchanged, all custom text CREF variable values stored in it remain intact.

  Note: If you want to delete only soft or hard properties, you can write a custom skill routine to override the CRefer default settings.

Understanding CRefer Output

Identifying Inputs and Outputs

There are two different ways in which you may find the direction information about a signal in the schematic:

1. **Input/Output Arrows** – If you have already cross-referenced a design, then CRefer adds direction characters (the input signals are represented by the < sign, and the output signals are represented by > sign) to the cross references. This information represents a quick way to identify inputs and outputs. If you select the *Omit Input/Output Arrows* check box to omit input/output arrows, then CRefer will not add this information.
2. **Hierarchical Arrows –** In a hierarchical design, a signal may be going up or down a hierarchy. If the signal is going up in hierarchy, then CRefer assigns the signal ^ as direction signal, and if the signal is going up in hierarchy then CRefer assigns the signal v as direction signal. If you select the *Omit Hierarchical Arrows* check box to omit hierarchical arrows, then CRefer will not add this information.

While CRefer makes it easy for you to identify the input or output information about a signal, it requires your help to determine this information correctly. CRefer needs the input/output information to display the direction of signals, and to place the cross references at the correct locations. Based on whether the signal is input or output, CRefer will place an XR string at the left or the right end of wire, respectively.

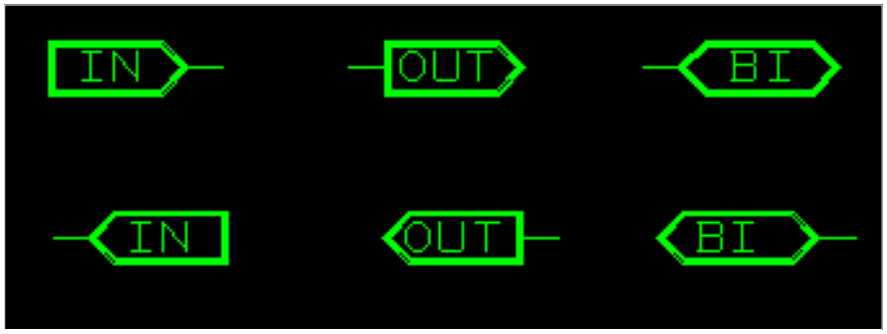
Note: It is strongly recommended that the direction information about a symbol be supplied at the time of schematic entry for accurate cross-referencing.

You can define the direction of a signal by attaching an OFFPAGE flag body to the signal. Flag bodies and ports are commonly used in schematic standards to indicate when the origin or destination of a signal is on another page of the schematic. Different versions of symbol bodies are used to define the signal as input, output, or bi-directional. If a signal has a flag body or port, CRefer uses the information contained in it to determine the direction of the signal.

When CRefer encounters a signal that does not have a flag body or port, it reads the *chips.prt* file for the component to which the signal is connected to obtain information about the direction of the component's pins. Based on the direction of the pin to which the signal is connected, the signal is marked as input, output, or bi-directional.

By default, CRefer understands the OFFPAGE, PORT, and FLAG bodies from the Cadence *standard* library. You can also create custom offpage flag bodies. However, if you add any flag body or port in the schematic, specify it in the cref.dat file to ensure proper cross referencing.

The six standard Cadence supplied versions of the OFFPAGE flag body are shown below:



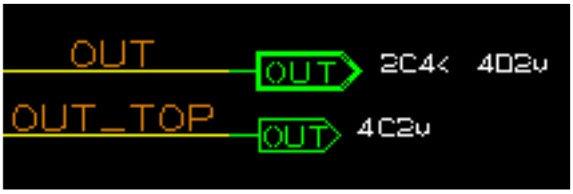
You can also add the OFFPAGE flag body to another library on your system.

Note: If you change the parts in the *standard* library, or add parts to it, be sure to keep a copy of the flag bodies at another location. Future Cadence library updates may delete any new or changed bodies in the library.

Sample Signals Labeled with Cross References

Example 1

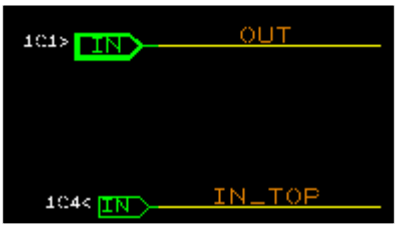
The following figure displays a drawing that has two signals annotated by CRefer.



1. The signal `OUT` appears twice, once as input at the zone C4 on page 2, and once representing a hierarchical signal going down at the zone D2 on page 4.
2. The signal `OUT_TOP` is a hierarchical signal, which appears at the zone C2 on page 4.

Example 2

The following figure displays a drawing that has two signals annotated by CRefer.



1. The signal `OUT` appears as output at the zone C1 on page 1.
2. The signal `IN_TOP` appears as input at the zone C4 on page 1.

CRefer reads your schematics and then rewrites them to include the cross-reference information. The cross-reference information is generated as properties attached to the nets or pins names. By default, CRefer assigns soft properties. You can, however, specify that CRefer assigns hard properties.

Note: By default, CRefer tries to place cross references for all signals. However, if CRefer does not find enough space for writing the cross references, it makes the cross reference values invisible to avoid overlapping of CRefer properties. You can use the `show properties` directive in Concept HDL to temporarily make the invisible cross references visible.

Signals that are Not Cross-Referenced

CRefer may not assign a cross reference to a signal if:

- The signal is unique.
- The input/output type of the signal cannot be determined because it is not attached to an OFFPAGE flag body or a port, or if the signal is attached to a horizontal wire with an unattached end.

When a signal cannot be cross-referenced, it receives an invisible property named $XRERR with the value as IOTYPE?. If the *Ignore Inputs Only Signals* check box is selected, then the $XRERR property may also be assigned the value NODRIVE? If the *Show Warnings for Unique Signals* check box is selected, then the $XRERR property may also be assigned the value UNIQUE?.

Note: If there is an invisible signal in the schematic, then an invisible cross reference is created for it.

To check why a signal was not cross-referenced, verify its property value. You can perform the following procedure:

1. Start Concept HDL and edit the schematic.
2. Group the invisible $XRERR properties by using one of the following 2 methods:
   - Select *Group* > *Create* > *By Expression*. This will display the Pattern dialog box in which enter $XRERR and click *OK*.
   - Enter the FIND $XRERR command at the Console Window.

   The properties are grouped and assigned the name ‘A’. If you have already created some groups in the current Concept HDL session, the group might have a different name.
3. Make the property values visible by using one of the following 2 methods:
   - Select *Group* > *Show Contents [A]*.
   - Enter the DISPLAY VALUE “A” command at the Console Window.

   Note: A is the group name assigned by the FIND command.

Use the Next command to automatically zoom in on each property.

Reference Information

CRefer Text Reports

CRefer generates four text reports: BaseNets report, NetsByPage report, Synonyms report, and CrefParts report. In addition, a CRefer error report - creferror.dat is generated. These reports are added in a separate view named *rptcref_1* view in the top-level cell for the current project. You can view the reports in a text editor.

BaseNets Report

The BaseNets report contains signal cross-reference information grouped according to each occurrence of the design cells. To view the report, open the `basenets.txt` file located in the *rptcref_1* view.

Note: CRefer puts the location of each signal and its synonyms in the reports. However, in case a design has buses split into multi-bit vectors, the report will list both the signal and its super-set as its synonyms.

NetsByPage Report

The NetsByPage report contains the list of nets and their base nets grouped by page. To view the report, open the netsbypage.txt file located in the *rptcref_1* view.

Note: BaseNets and NetsByPage reports include the direction characters with the signal and location information.

Note: If you have a bus at the top level that is split at the lower level, then the Basenets and NetsByPage reports will display only the Most Significant Bit (MSB) in the synonym list of the bus at the top level. For example, the bus `A<2..0>` at the top level will show only the `A<2>` bit as its synonym. At the lower level, however, the cross referencing information will be generated for all bits.

Synonym Report

The Synonym report contains information about the synonyms corresponding to each base signal along with their location, zone, and direction information. This report groups the nets and the basenets in a design by the design sheets.

Crefparts Report

The CrefParts report contains information about all unit cross references for the entire design. The information includes the path property attached to the cell, the symbol name, and the cross references.

In PSD 14.0, the crefparts report did not include information for the replicated hierarchy in the design. In PSD 14.2, CRefer reads all the instances of the cells in both replicated and non-replicated hierarchy to generate complete reports.

Cref Error Report

The Cref error report is generated when CRefer encounters any warning. This report is useful for debugging purposes. You may check this report by opening the `creferror.txt` file in the *rptcref_1* view under the top-level design.

Note: The cref error report is a subset of the Cref log file.

Schematic Reports

CRefer generates two schematic reports: signal cross reference report and part cross reference report. The signal cross reference report lists all signals and the list of all places where the signal is being cross-referenced. The part cross reference report lists all parts in the design and their locations.

To create signal part reference and part cross reference schematic reports, select the *Generate Schematic Reports* check box in the Cross Referencer Options - Reports tab. When you select the *Generate Schematic Reports* check box, the *Create Separate View for Schematic Reports* radio button is selected by default. CRefer creates the two schematic reports in a separate view named *crefout*, under the root design. This view contains two pages, one each for signal and part cross reference reports.

Cref Data File

The cref data file is used for:

- Creating the Cref Data File for Page Borders on page 464
- Creating Custom Offpage I/O Flag Bodies on page 465
- Suppressing Cross Referencing of Signals on page 465
- Creating Cross References for Power Signals on page 465

Sample Cref Data File

The following is a sample cref.dat file for an A SIZE PAGE Border:

```
pagename A SIZE PAGE
version 2
lowerleft (-3650, 475)
upperright (-50, 4950)
xmark 1 -600
xmark 2 -1525
xmark 3 -2500
xmark 4 -3425
ymark A 850
ymark B 2225
ymark C 3300
ymark D 4625
pagenumber (-425, 50)
pagenote (-1500, 50) Eng Name
pagenote (-1450, 200) date
```

The directives used in the sample are:

`pagename`	This directive identifies the page border symbol name (the name that you would use with the `ADD` directive in Concept HDL).
`version`	This directive identifies the symbol version of the page symbol; you can omit this directive if the version number is 1.
`lowerleft`	This directive sets the lower left corner coordinate, in default units. To determine the coordinate, use the `SHOW COORDINATE` directive in Concept HDL and click at that point.
`upperright`	This directive sets the upper right corner coordinate, in default units. The coordinates for `lowerleft` and `upperright` should define the largest possible rectangle that fits on the sheet, taking into account the space occupied by the title box.
`xmark`	This directive specifies the horizontal coordinate of the cross reference key letters or numbers located along the top or bottom of the page. When you use the `SHOW COORDINATE` directive, click the mouse directly on the top of the number or letter; do not click on the boundary lines. For more information on how to determine coordinates in a schematic, refer determining coordinates.
`ymark`	This directive specifies the vertical coordinate of the key letters or numbers along the left or right side of the page. Click the mouse directly on the top of the number or letter when you use the `SHOW COORDINATE` directive; do not click on the boundary lines.

`pagenumber`	This directive specifies the coordinate where CRefer prints a page number for the signal and part cross reference summary sheets. You can omit this directive if you do not want the pages numbered.
`pagenote`	This directive prints a text note anywhere on the page. You can use as many notes as you wish.

The Cadence B SIZE PAGE border is given below:

```
pagename B SIZE PAGE
version 1
lowerleft (-8125, -125)
upperright (125, 5125)
xmark 1 500
xmark 2 1500
xmark 3 2500
xmark 4 3500
xmark 5 4500
xmark 6 5500
xmark 7 6500
xmark 8 7500
ymark A 1000
ymark B 2200
ymark C 3325
ymark D 4500
pagenumber (-425, 50)
pagenote (-2000, 50) Eng Name
pagenote (-500, 250) date
```

I/O Types

By default, CRefer writes the characters indicating the I/O type of the signals. The following table lists the different characters used along with their descriptions.

Character	Description
<	The signal is an input Signal
>	The signal is an output signal
<>	The signal is an inout (Bidirectional) signal
^	The signal is going up in the hierarchy.
v	The signal is going down in the hierarchy.

Page Numbering

CRefer assigns a page number to each schematic sheet in the design hierarchy. The page numbers used in cross references are based on these page numbers. CRefer writes these numbers at the top left corner of the sheet. In case of flat designs, the assigned page numbers are the same as the Concept HDL sheet numbers.

Support for Occurrence Property Data

CRefer version 14.0 supports occurrence property data. This data comes from the Occurrence Property File (OPF). The OPF is a database that stores properties for all objects (instances, pins, and nets).

The OPF property data is annotated on the top of the schematic. The information from the OPF is particularly useful for designs that have hierarchical modules or read-only modules.

Depending upon whether you are generating the flattened schematic view or not, CRefer handles OPF data in two ways.

When a flattened schematic view is generated, by selecting the *Generate Flattened Schematic* check box, a separate flattened view, *schcref_1*, is created. CRefer reads the Occurrence properties from the OPF file and annotates them on the flattened schematic as schematic properties.

When a flattened schematic view is not generated, CRefer properties are written as schematic properties and OPF properties are read by Concept HDL from the props.opf file in the OPF view.

Note: In case an OPF property already exists as the schematic property, CRefer will overwrite the winning value of the property. A message to that effect will be written into the Cref log file. For more information about OPF and how a winning value is calculated, see *Packager-XL Reference*.

Support for Concept HDL Custom Variables

Custom variables are special variables, which are supported by Concept HDL. You can use these variables for intelligent plotting of cross-referenced schematics. For example, using these variables you can place page information such as ‘This is page 1 of 24’ on the cross-referenced schematics. You can also use custom variables to store information such as the company name and author name.

Adding CRefer Custom Variables

There are 7 variables, which are specifically useful for CRefer - 5 of these existed in PSD 14.0 and 2 new variables have been added in PSD 14.2. For example, the `CREF_TO_LIST` variable defines where the pages for the block are located in the cross-referenced flattened design. Similarly, the `CREF_FROM_LIST` variable defines where the pages are coming from in a design.

To add any CRefer variable, you need to first define a custom text and include the variable. After defining the custom text, you attach it to an object on the schematic. If you are adding the `CREF_TO_LIST` variable or the `CREF_FROM_LIST` variable, then the ideal place to add the variable is at the top or near the block for which you want to find cross-referenced page data. For other CRefer variables, the ideal location is the page border.

Note: CRefer specific custom variables will be substituted when you run CRefer with the *Generate Flattened Schematic* check box selected.

The list of available CRefer variables is displayed in the following table:

Variable Name	Description
CREF_TO_LIST	Defines where the pages for the blocks are located in the cross-referenced flattened design.
CREF_FROM_LIST	Defines where the pages in a flattened design came from in the original design.
CREF_ORIG_DESIGN_NAME	Defines the original design name
CREF_ORIG_PAGE	Defines the original page number
CREF_ORIG_VIEW	Defines the name of the original view
TOTAL_DESIGN_SHEETS	Lists the total number of pages in the Concept HDL schematic
CURRENT_DESIGN_SHEET	Lists the sheet number of the current page in the schematic

Note: The `TOTAL_DESIGN_SHEETS` and `CURRENT_DESIGN_SHEET` variable are new variables available in PSD 14.2. For more information about using these variables, see Design Sheet Variable Support on page 491.

Example of Adding a CRefer Variable

You will add the CRefer specific variable CREF_TO_LIST to the top of module named DOWN in the design CREF1. The custom text using the variable CREF_TO_LIST will display the message “This block goes to page <CREF_TO_LIST>”.

Steps

1. Choose *Text > Custom Text* in Concept HDL to display the Custom Text dialog box.
2. In the *Format string* field, enter the following text: “This block goes to page”.
3. Select the <*CREF_TO_LIST*> variable in the *Variables* list.

   The *Display string* is automatically appended.
4. Click *OK*.

   The custom text is attached to a cursor.
5. Move the cursor at the top of the block down and click to attach the custom text to this block.
6. Click again to place the custom text on the schematic.
7. Right-click and press *Done* to complete the operation.

The custom text ‘This block goes to page <CREF_TO_LIST>’ attached to the block DOWN and when you cross reference the design, the block DOWN displays the following text: This block goes to page 3” where 3 is the value of the <CREF_TO_LIST> variable.

Performing To/From Property Annotation

CRefer performs to/from property annotation to specify where the pages in a flattened design come from the original design and where the pages from the original hierarchical design are included in the cross-referenced flattened design.

To perform to/from property annotation, CRefer uses CREF_TO_LIST and CREF_FROM_LIST custom variables.

Note: To/from property annotation is performed only when a new flattened view (*schcref_1*) is created.

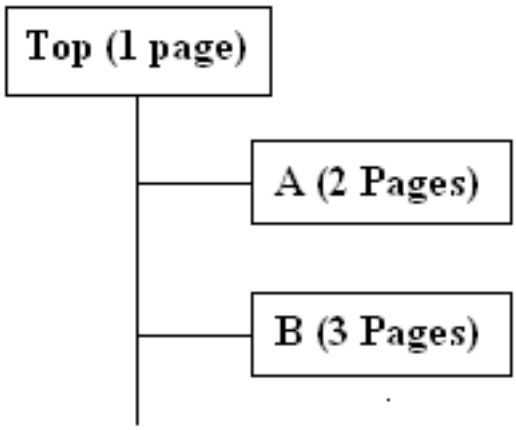
Steps for Performing To/From Property Annotation

Define and attach custom text for the CREF_TO_LIST and CREF_FROM_LIST variables in the schematic through Concept HDL. For more information about attaching custom text, see Concept HDL Help.

1. CRefer calculates the values of custom variables in the custom text and provides it to Concept HDL.
2. Concept HDL updates the custom text with the values received from CRefer and annotates the properties on the schematic.

Example

Assume you have the following hierarchical design named Sample:



The Sample design has one page at the top level (TOP). The TOP page contains two blocks A and B of two and three pages respectively. After cross-referencing, the design will consist of six pages, where Page 1 corresponds to the block TOP, pages 2 and 3 correspond to the block A, and pages 4 to 6 correspond to the block B.

Under (attached to) the symbol on page 1 for block A is annotated a property:

```
CREF_TO_List = Pages 2, 3
```

While on the page border of pages 2 and 3 is annotated another property:

```
CREF_FROM_List = 1B3 – 1P
```

Where 1P is the instance name (for block A) and 1B3 is the cross reference for the hierarchical symbol.

Similarly under the symbol on page 1 for block B is annotated a property:

```
CREF_TO_List = Pages 4, 5, 6
```

While on the page border of pages 4, 5 and 6 is annotated another property:

```
CREF_FROM_List = 1C7 – 2P
```

Where 2P is the instance name (for block B) and 1C7 is the cross reference for the hierarchical symbol.

Note: For replicated and read-only blocks, CRefer calculates the value of the CREF_TO_LIST and CREF_FROM_LIST custom variables and annotates them to the cref.opf file as CDS_CREF_TO_LIST and CDS_CREF_FROM_LIST variables, respectively.

Note: If there are CRefer-specific custom variables annotated to the page border, then CRefer annotates that variable to the canonical name of the schematic in the cref.opf file.

Example

Assume you have the following custom text:

```
CRefer from list is <CREF_FROM_LIST>
```

on page 1 of the schematic MUX, then CRefer will substitute the CREF_FROM_LIST property on the canonical name for page 1 of the schematic MUX. The property will have the following value:

```
I1-3C5
```

where, I1 represents the first instance of the MUX block, 3 represents the page number of the instantiating block, and C3 represents the zone coordinate of the instantiating block.

Design Sheet Variable Support

CRefer in PSD 14.2 supports two new custom variables named TOTAL_DESIGN_SHEETS and CURRENT_DESIGN_SHEET. These variables are also supported by Concept HDL.

- The TOTAL_DESIGN_SHEETS variable lists the total number of pages in the Concept HDL schematic. While calculating the TOTAL_DESIGN_SHEETS value, the number of pages in all modules of the design are taken in account.

  Note: The TOTAL_DESIGN_SHEETS value is the same as the total number of pages generated by CRefer when it cross references a design by creating a flattened view of the base schematic.

- The CURRENT_DESIGN_SHEET variable lists the sheet number of the current page in the schematic.

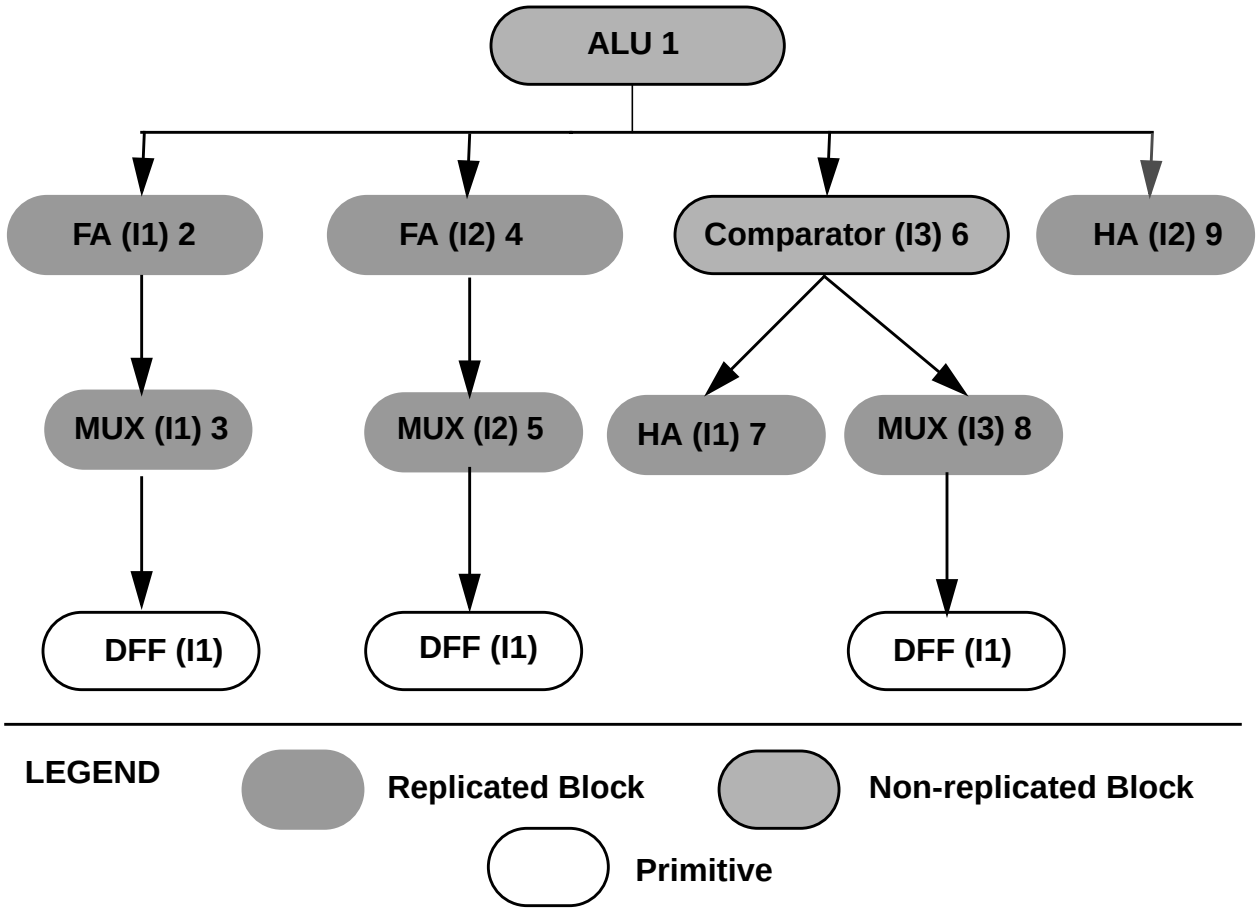
Note: Whenever you add or delete pages, or perform module ordering, the value of the TOTAL_DESIGN_SHEETS and CURRENT_DESIGN_SHEET variable changes. However, each time a change occurs in the schematic, Concept HDL does not reevaluate the value of the TOTAL_DESIGN_SHEETS and CURRENT_DESIGN_SHEET variables. Therefore, it is important that you should cross reference the design to get the updated values of the TOTAL_DESIGN_SHEETS and CURRENT_DESIGN_SHEET variables.

Note: You can easily search for any custom text variable by selecting it in the *Variables drop-down* list in the Custom Text dialog box in Concept HDL.

Example of Design Sheet Variables

Assume you have assigned the custom text ‘This is sheet <CURRENT_DESIGN_SHEET>’ on the MUX schematic in the DFF (I1) block as shown in the following figure.

Figure 17-5 ALU Hierarchical Design



Naming Convention in Block: Block name (instance name) Design current sheet value

Also assume that all blocks in the ALU design are single page blocks that have not been reordered. As the design is traversed in the depth-first order, the custom text ‘This is sheet <CURRENT_DESIGN_SHEET>’ on the MUX schematic in the DFF (I1) block will result in the following OPF properties at the @alu_lib.alu(opf).

```
@alu_lib.alu(sch_1):page1_i1@alu_lib.fa(sch_1):page1_i1@alu_lib.mux(sch_1):page1
_i1@alu_lib.dff(sch_1):page1
This is sheet 3

@alu_lib.alu(sch_1):page1_i2@alu_lib.fa(sch_1):page1_i2@alu_lib.mux(sch_1
):page1_i1@alu_lib.dff(sch_1):page1
This is sheet 5
```

If you have assigned the TOTAL_DESIGN_SHEETS variable as part of custom text to any block in the schematic, then that variable would had the value 9.

Important

The custom variable functionality in Concept HDL in PSD 14.0 was limited. In PSD 14.2, Concept HDL displays properly substituted custom variables on the schematic. For non-replicated hierarchy, Concept HDL always substitutes the CRefer-specific custom variables. However for replicated hierarchy, Concept HDL substitutes the CRefer-specific custom variables ONLY in the Occurrence Edit mode. Concept HDL is not supporting adding the custom text functionality in the Occurrence edit mode.

Updating Custom Text Variables for Page Numbers

You need to update the CURRENT_DESIGN_SHEET and TOTAL_DESIGN_SHEETS custom text variables for page numbers to ensure that the schematic page displays the correct page number. This updating needs to be done in the following cases:

- If you have modified the design by adding or deleting pages or blocks.
- When you add custom text variables for page numbers on a schematic page, the name of the variable is substituted by the page number. If the schematic page continues to display the variable name instead of the page number, you need to update the variable. For example, if you add the CURRENT_DESIGN_SHEET custom text variable in the 20th page in a hierarchical design, the schematic page may display <CURRENT_DESIGN_SHEET> instead of 20.
- When you renumber schematic pages using the page renumbering commands, the schematic pages may not display the correct page number.

Note: When you perform module ordering, cross-referencing, or plotting of the design, the custom text variables are updated automatically.

To update the custom text variables for page numbers

➤ Choose *Text > Update Sheet Variables*.

The custom text variables for page numbers on all pages in the design are updated to display the correct page number.

Placeholder Support

To ensure that CRefer can optimize the placement of cross references in a design, you can add placeholders on the schematic.

perform the following procedures:

- Adding Placeholders on Ports or Offpage Symbols on page 495
- Editing of Invisible Placeholders on page 497
- Managing Changes to the standard Library on page 17-498

Note: In many firms, librarians may be responsible for creating placeholder support for symbols.

Adding Placeholders on Ports or Offpage Symbols

CRefer will only place annotations for the signals that have a $XR<n> placeholder attached on the port or offpage symbol. Therefore, add placeholders on all ports and offpage symbols. To add a placeholder to a port or offpage symbol:

1. Open the Attribute form.
2. Assign the following placeholder property to the port or offpage symbol:

   ```
   $XR0=?
   ```

Caution

When putting placeholders, start from the `$XR0` property. Do not use any other property for defining placeholders.

3. Repeat step 2 to assign more placeholders. Add the placeholders in a contiguous manner as `$XR0`, `$XR1`, `$XR2`, and so on. For example to create the second placeholder, define the following property to the port or offpage symbol:

   ```
   $XR1=?
   ```

   Note: The number of placeholders in a design depend upon the nature of your design. Since different designs may have different number of cross annotations on these ports or offpage symbols, it is recommended that `version 1` of your symbols has enough placeholders to allow proper cross referencing. When CRefer runs out of placeholders, it dumps all the remaining cross references on the last place holder. The following message is also generated:

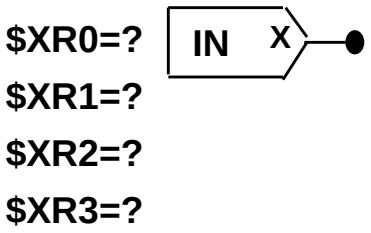
   ```
   Signal <signal_name> at <location_of_SIG_NAME> required
   <number_of_required_placeholders> placeholders on
   <name_of_the_attached_plumbing_body>.
   ```

Note: If you have both ports and offpage symbols connected to the same point of a signal and if you have selected the *Distinguish Between Ports and Offpages* check box in the Cross Referencer Options - Content Tab, then cross references will be attached to either port or offpage symbol, but not to both. To ensure that cross references are attached to both ports and offpage symbols, connect the port and offpage symbol to the signal at different points.

Recommended Steps

1. Add all ports and offpage symbols to the `cref.dat` file located at `<your_install_dir>/share/cdssetup/creferhdl`. For more information about the `cref.dat` file, see Cref Data File on page 17-484.
2. Create different versions of symbols that allow both vertical and horizontal stacking. This will ensure that a designer can switch between vertical and horizontal stacking by versioning the instances in the schematic. You can also assign different number of placeholders in different versions of symbols.

Figure 17-6 Vertical Stacking of Placeholders



3. Use the appropriate alignment - CRefer will use the standard Concept HDL alignment for writing cross annotations on placeholders. It is recommended that you left-align the placeholders for all `OUT` ports and right-align the placeholders for all `IN` ports. You should select a suitable alignment for other ports and offpage symbols. Figure 17-7 on page 497, Figure 17-8 on page 497, and Figure 17-9 on page 497 provide an idea about why appropriate alignment is necessary. Notice that left-alignment of placeholders is causing the cross references to overlap on the `IN` symbol. However, right-alignment of placeholders causes proper placement of cross references.

Figure 17-7 IN Port Connected to MUX Block: Before Property Substitution

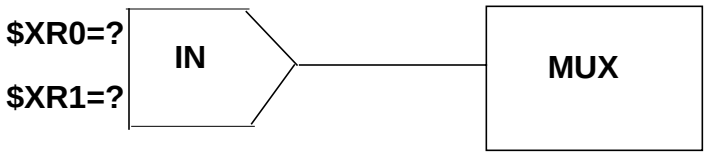


Figure 17-8 IN Port Connected to MUX Block: After Left-aligned Property Substitution

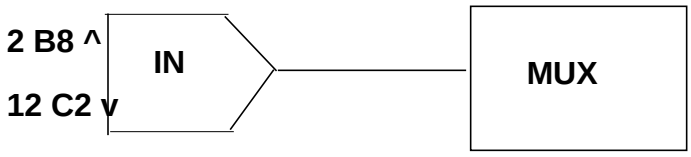
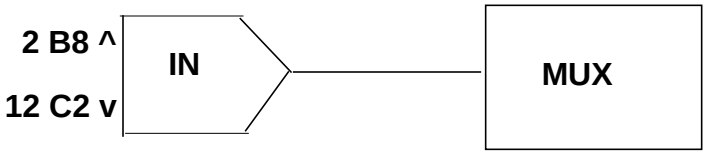


Figure 17-9 IN Port Connected to MUX Block: After Right-aligned Property Substitution



4. Make the placeholders invisible - If you have made placeholders on ports, or offpage symbols as visible, then on cross referencing the design in PSD 13.6 version, you will see the ? sign in all placeholders that are not substituted by CRefer. To avoid this problem, you should make placeholders on ports or offpage symbols invisible by selecting *Group > Property Display > Invisible* in Concept HDL.

Editing of Invisible Placeholders

To edit an invisible placeholder, you need to perform the following steps:

1. Create a group of all placeholders by selecting *Group > Create > By Expression* in Concept HDL.
2. Type `$XR*=?` and press Enter.

The group is created.

3. Make the placeholders visible by selecting *Group > Property Display > Value* command in Concept HDL.
4. Edit the placeholders and move them as required.
5. Select the group of placeholders and make them invisible by selecting *Group > Property Display > Invisible* in Concept HDL.

Managing Changes to the standard Library

If you make changes to any symbols in the Cadence `standard` library, such as adding placeholders to ports or offpage symbols, you may lose those changes when QSRs or updates to the `standard` library are installed.

It is recommended that you make a local library of all symbols that you customize in the `standard` library, and use that library. Avoid making changes to symbols in the `standard` library directly.

18

Archiving a Project

This chapter contains the following information:

- Creating a New Archive on page 500
- Opening an Archive on page 502

The following figure illustrates the location of this phase in the Front to Back flow.

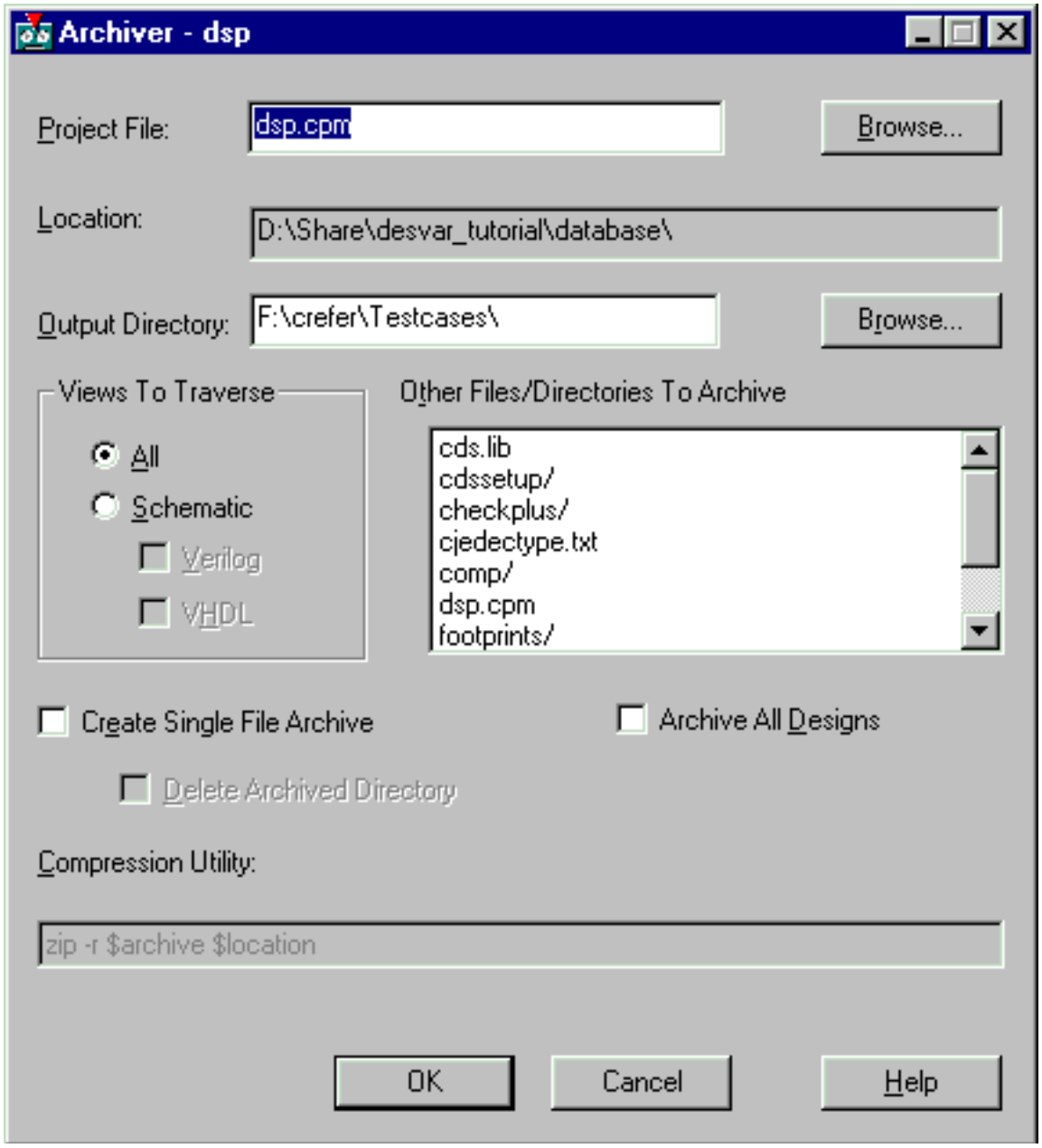
Archiver Overview

Concept HDL supports reference-library paradigm for schematic design. You can create a design by referencing parts from multiple libraries. This approach allows the independent maintenance of libraries, and consistent use of parts in all designs. If parts are changed in the reference library, they automatically change in all designs that references them. However, you may like to make a snapshot of your design at a particular point in time. This snapshot will contain all the parts used in your design locally. Such a snapshot is called an archive in Concept HDL terminology. You can use Archiver to create an archive of your design or to extract the design from a previously created archive.

Archiver also has the Archopen feature which connects to any new reference library set and lets you determine whether any differences exist between the new reference libraries and the libraries you had archived. If any differences are found, you can archive the new reference library set.

Creating a New Archive

To create a new archive, you have to open the project you wish to archive in Project Manager. Choose *Tools > New Archive* to access the Archiver dialog box. You can also double-click on *archiver.exe* in Windows Explorer to run Archiver.



The *Project File* field displays the selected project you wish to archive. The project you open in Project Manager appears by default. If you want to archive another project, you can specify

the project file by clicking the *Browse* button. Specify the path to the output directory where Archiver will save the output.

Archiver traverses all views of the design by default. If you select the *Schematic* option button, Archiver traverses only the schematic view of your design. If you select *Verilog* or *VHDL* check box, Archiver archives only that views. See Example of Using Views To Traverse on page 501 for details.

The *Other Files/Directories To Archive* list box displays other files and directories contained in the current directory. You can select any of these elements for archiving them as well. This feature is particularly useful if you want to archive some additional documentation.

By default, Archiver creates a new directory named `<root design>_archive`. This directory contains the root design, the project file, the `cds.lib` file, the part table files, the files corresponding to all components in the design, and any other files or directories that you may have selected using the *Other Files/Directories To Archive* option.

You can specify that the entire archive be created as a single file. For this, select the *Create Single File Archive* check box. Depending upon your operating system, the *Compression Utility* field will display the compression command. The tar command is used in case of Unix or Solaris systems, and the zip command is used in case of Windows systems.

When you select the *Create Single File Archive* check box, the *Delete Archive Directory* check box becomes active. You can select this check box to delete the archived directory for the design.

If you want to archive all designs in the current directory, select *Archive All Designs* check box.

Finally, select *OK* to create the archive.

Example of Using Views To Traverse

Assume you have the root design `top`, which has two views `sch_1` and `vlog_model`. In the `sch_1` view, assume the bottom cell is instantiated and in the `vlog_model` view, the bottom1 cell is instantiated. This configuration now has two possible hierarchies, `top->bottom` (traversing `sch_1`) and `top->bottom1` (traversing vlog_model).

By default, the *Views To Traverse* option is set to *All*. This selection will cause Archiver to traverse both `sch_1` and `vlog_model` views and the 3 cells: `top`, `bottom`, and `bottom1` are archived.

Now if the *Views To Traverse* option is set to *Schematic*, only the top and bottom cells are archived. In all cases, all views of the cells are archived. Therefore, if bottom has a packaged

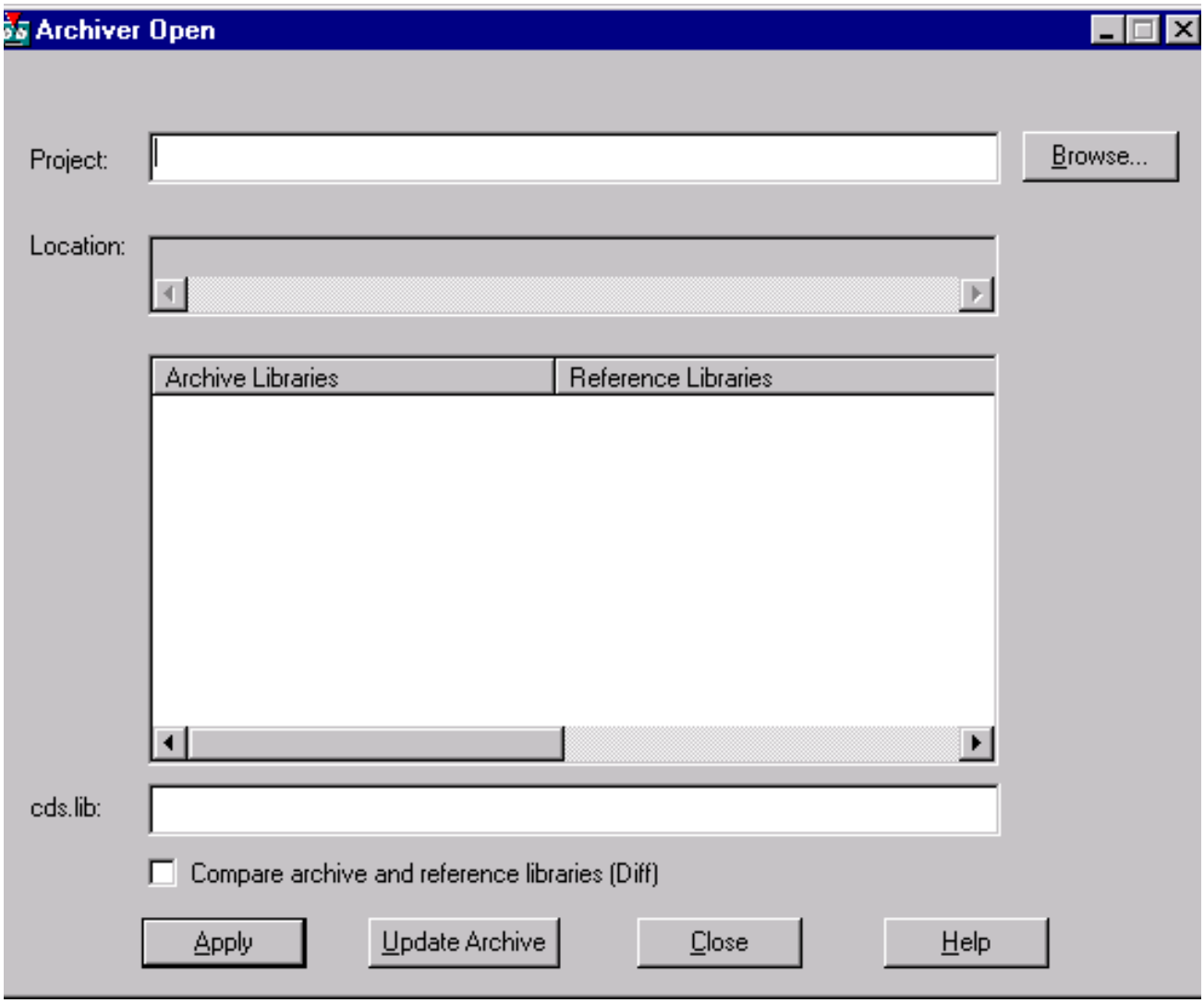
view, it will always be archived. In short, the *Views To Traverse* option only determines the traversed cells. It does not control which views get archived.

Opening an Archive

The Archiver Open feature lets you open an existing archive and connect to the current set of reference libraries. After connecting to the reference library set, Archopen displays the differences between the archived libraries and the current reference library set.

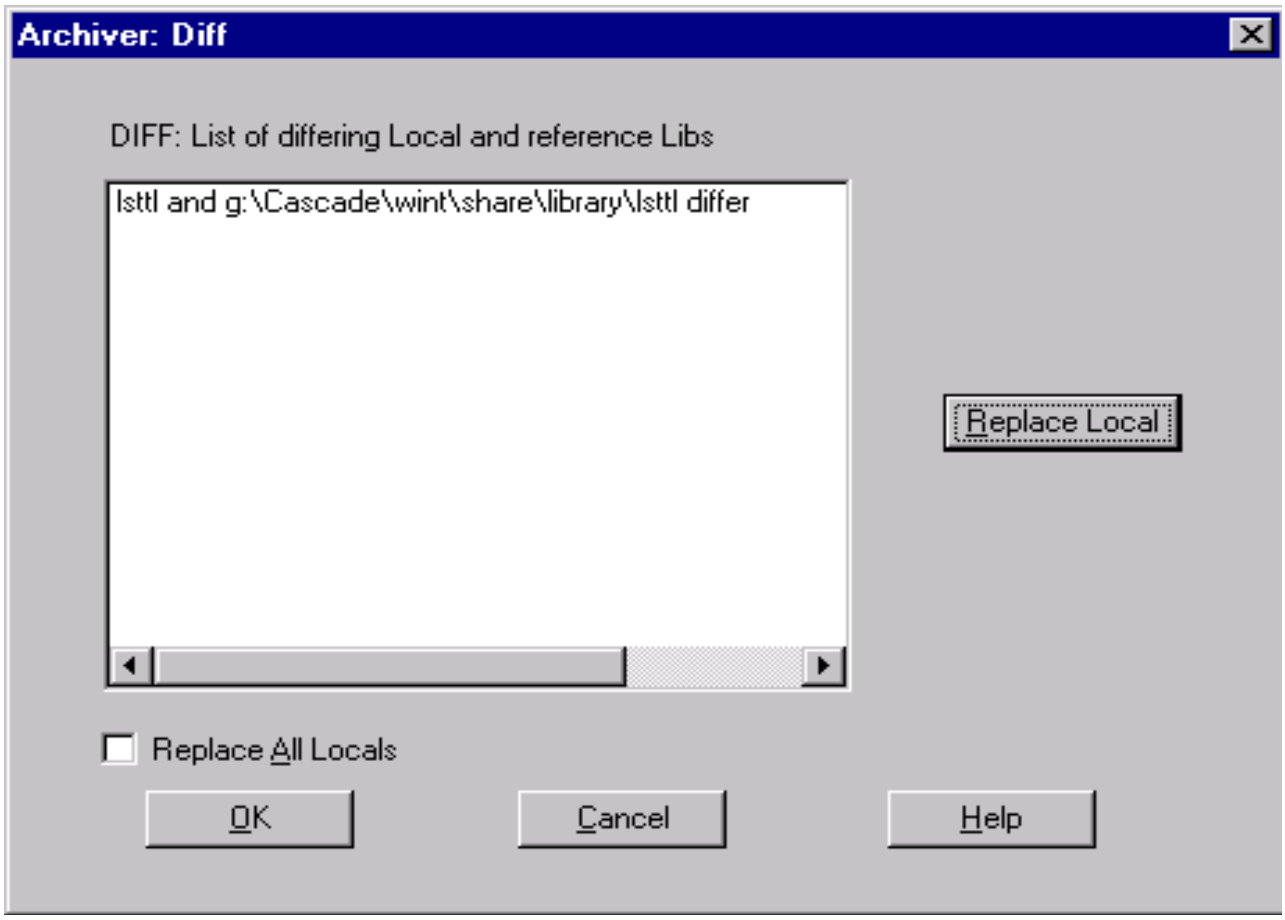
To open an archive, choose *Tools > Open Archive* in Project Manager (or double-click *archopen.exe* in Windows Explorer) to access the Archiver Open dialog box and specify the project to be opened. Archiver displays the list of libraries present in the archive and the corresponding reference libraries. Enter a different *cds.lib* file if you want to display a different

set of reference libraries. if you want to view a list of reference parts that are different from those archived select *Compare archive and reference libraries* check box.



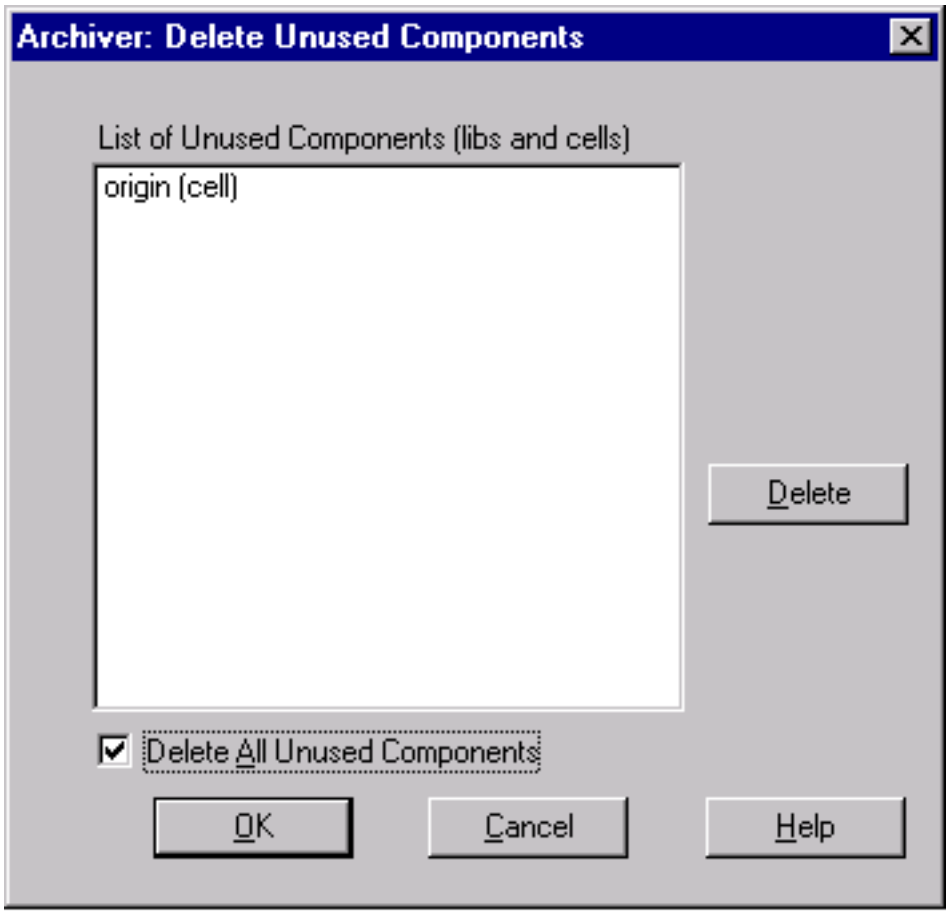
When you click *Apply*, and if you have selected *Compare archive and reference libraries* check box, Archiver compares the parts in the reference libraries with the parts in the archived

libraries and generates a list of differences. After viewing the differences, you can replace the archived parts with the ones that have been modified in the reference libraries.



When you click the *Update Archive* button, Archiver copies over any components that have been modified in the reference library into the archive. If you replace the archived components with components from the new reference library set in your design, some

components might no longer be used. You can delete these components using the *Delete Unused Components* dialog box.



19

Design Techniques

Introduction

This chapter introduces the three basic design techniques: flat, structured, and hierarchical. One of these three techniques may best meet your needs:

- Flat Design Technique

  The flat design technique is an efficient method for creating a design that is small and does not re-use portions of the circuitry. Flat designs are required for complete backannotation of the design and are more convenient for troubleshooting. Flat designs can include multiple drawing pages.

- Structured Design Technique

  The structured design technique allows abbreviated bus structures and minimizes the required number of parts and interconnections. Structured design techniques using the SIZE property support designs that use large bused signals, register depth, and memory depth.

- Hierarchical Design Technique

  The hierarchical design technique uses symbolic representations of circuitry for functions that are repeated throughout a design. Large designs that can be broken into functional modules or designs that re-use portions of circuitry can be efficiently created with a hierarchical technique.

Although all designs can be entered as flat drawings, choose the method most appropriate to your particular design. Concept HDL and the other system design tools are specially designed to operate efficiently with structured and hierarchical techniques.

Flat Designs

The flat design method is the most straightforward technique for creating a design with the Cadence system design tools. In a flat design, all parts on the drawing come from Concept

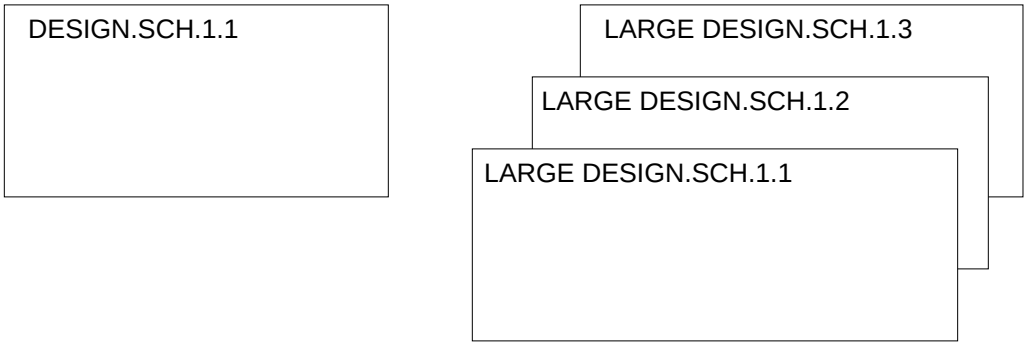
HDL or user-defined libraries and are one-to-one logical representations of the physical parts. The entire interconnecting wiring within the design is entered pin-to-pin.

Flat designs are best suited for small designs that do not have sophisticated bus requirements and do not re-use portions of circuitry. Also, if the design must be completely backannotated with pin and physical location numbers, a flat drawing is required.

Creating a Flat Design

Both single-page and multiple-page flat drawings can be created with Concept HDL and processed by the Cadence design analysis programs.

Figure 19-1 Single and Multiple Drawing Pages



Some designs are small enough to fit on one page of a drawing.

To create a single-page design,

1. Specify the drawing name with *File > Open*.
2. Use Concept HDL to draw the design on the screen.
3. Use *File > Save* to store the design on the disk.
4. Use *File > Export Physical* to package the design.

If the drawing is too large to fit on one page, create a multiple-page drawing.

To create a multiple-page drawing

1. Specify the drawing name with *File > Open* and create page1 of the design.
2. Use *File > Save* to save page 1.
3. To begin page 2 of the drawing, use *File > Edit Page/Symbol > Insert Page*.

4. Use *File > Save* to save page 2.
5. Create subsequent pages of the drawing in the same way.

All pages of a multiple-page design have the same drawing name. The system links all drawings with the same name. If the names are different, each page is treated as a separate drawing.

Give signals that cross page boundaries the same signal name on subsequent pages. Signals with the same name have an implicit connection, even if they appear on different pages. For example, the signal SYSTEM CLK on pages 1 and 3 has the same effect as being on the same page with both instances wired together.

Concurrent Engineering of Flat Designs

Concept HDL allows teams of designers to simultaneously work on different schematic pages in a flat design. When a designer is editing a page in the design, Concept HDL locks the page and does not allow other designers to modify the page. For more information on page locking, see What is Page Locking? on page 58.

On UNIX, each designer must set `umask` settings so that all other designers have write permissions to all the files in the schematic view of the design. For example, if each designer sets `umask` to `0` before starting Concept HDL, other designers will be able to simultaneously work on different schematic pages of the flat design.

Cadence recommends that you disable netlisting of the design if multiple designers are working on different schematic pages in a flat design. To disable netlisting of the design, do the following:

1. In Concept HDL, choose *Tools > Options*.
2. The *Concept Options* dialog box appears.
3. Select the *Output* tab.
4. Deselect the *Create Netlist* check box.

After all the designers have completed their changes, you can enable netlisting of the design and save the design.

Considerations of Flat Designs

Keep these considerations in mind when you create a flat drawing:

- Flat designs take longer to create and process than structured and hierarchical designs.

- Flat designs tend to be cluttered and hard to read unless special care is taken to organize and layout the design.
- Troubleshooting errors in a large, multiple-page flat design is time-consuming and difficult.

Structured Designs

The structured design method facilitates the entry and analysis of sophisticated designs that make use of bused signals, memory and, register depth. A structured design minimizes the number of interconnections and parts on the schematic.

Creating a Structured Design

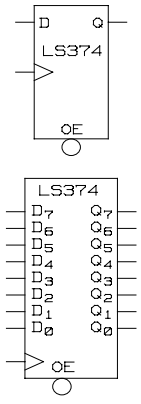
You use Concept HDL commands to enter and store your drawing. The main difference between a structured design and a flat design is the use of special library parts and the SIZE and TIMES properties.

SIZE Property

The SIZE property is attached to a symbol and is used to specify the width of pin names and signal names and to define size expansion.

For example, there are two versions of an LS374 octal register in the LSTTL library. Version 1 is a one-bit slice of the part. It accepts a vectored D input and produces a vectored Q output. Version 2 is the full-chip representation of the LS374 with all eight input and output bits explicitly shown.

LS374 symbol
Versions 1 & 2

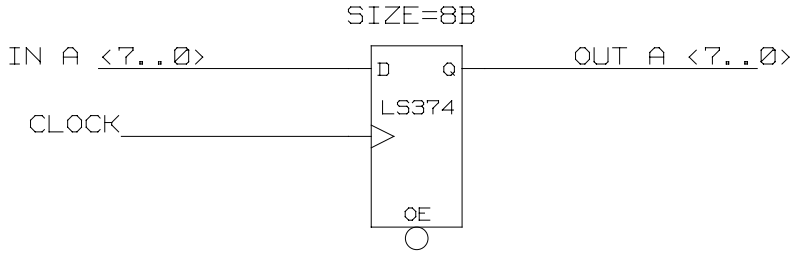


Version 1 is sizeable, which implies that you can specify the number of bits the part can represent. Library parts are generally developed with version 1 being sizeable. The *show vectors* command displays the pin names of a selected part allowing you to verify that a part is sizeable.

You attach the SIZE property to version 1 of the LS374 part to define the number of bits the pins D and Q represent. The signal syntax for bus notation is used to specify a range of bits for the input and output signals.

Figure 19-2 illustrates how you can use version 1 of the LS374 part in a structured design. In this example, the number of bits is set to 8 (SIZE = 8B) any number of bits can be specified to meet your requirements.

Figure 19-2 Using the SIZE Property To Structure LS374



Version 2 of LS374 is a flat representation of the part. Each pin on the drawing represents a pin in the physical package.

Figure 19-3 Using Version 2 of LS374

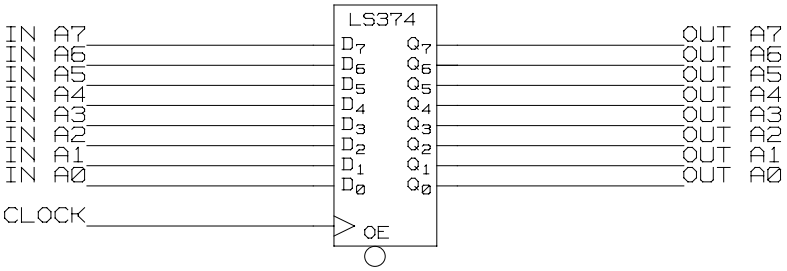
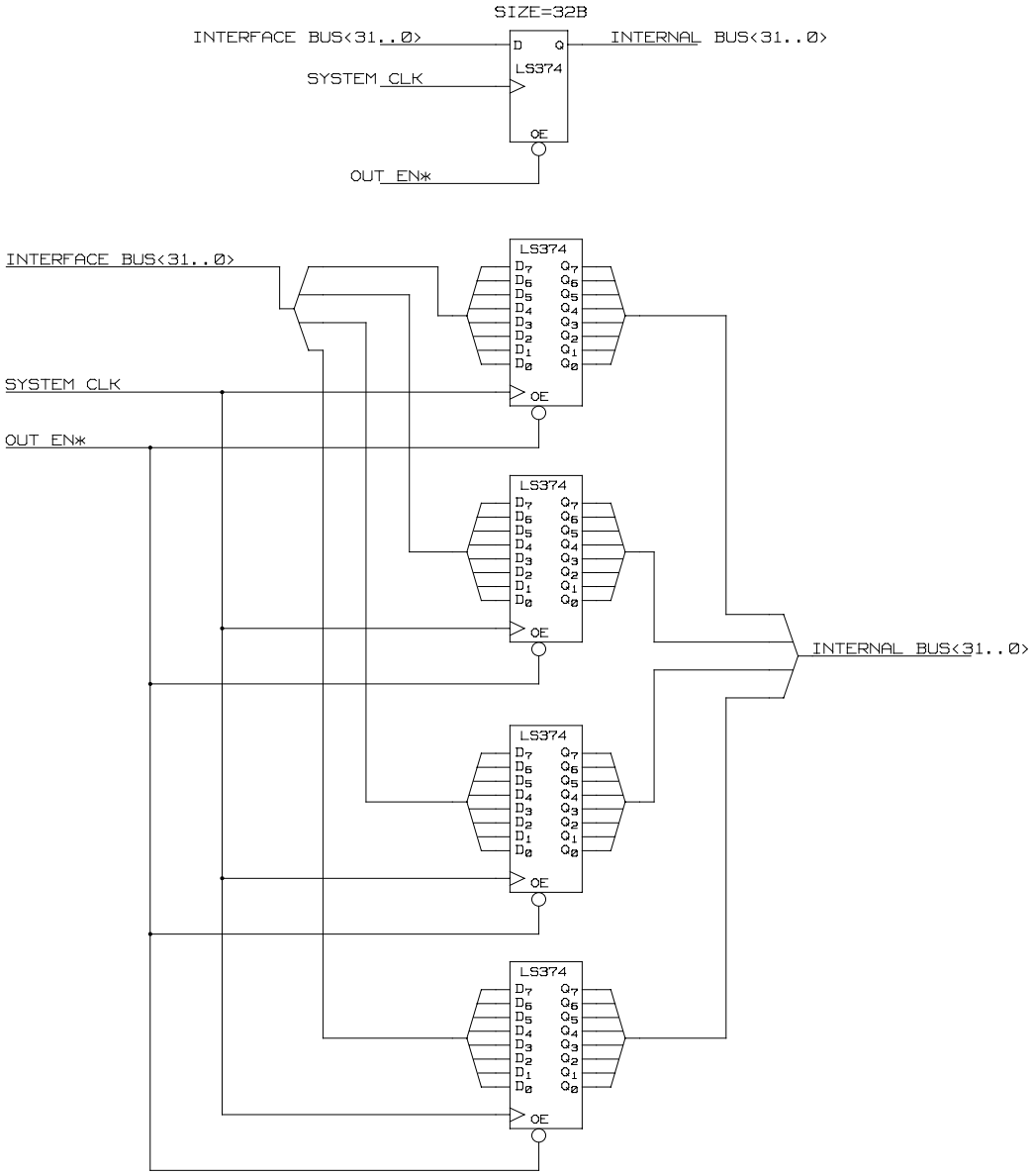


Figure 19-4 on page 512 illustrates the difference between the structured design and flat design techniques. Using the SIZE property can greatly minimize the number of parts and interconnections required. Also, you can avoid many possible entry errors.

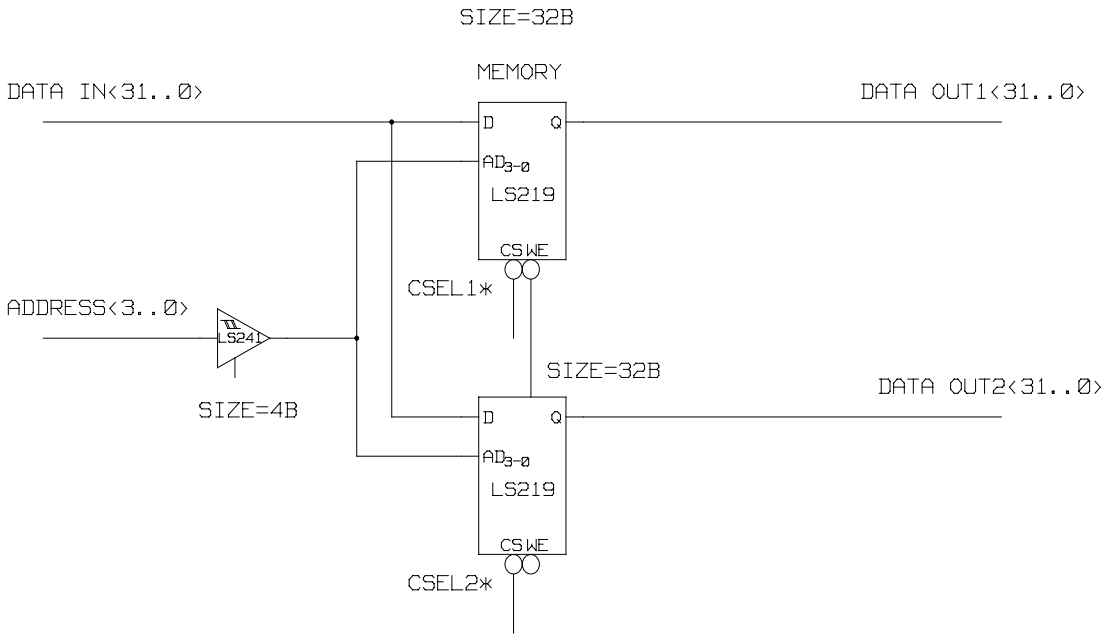
Figure 19-4 Structured and Flat Design Techniques



TIMES Property

The TIMES property is used with the SIZE property on structured designs. TIMES allows you to create your structured design to data book specifications. TIMES can be used in cases where the SIZE property causes loading errors. For example, in Figure 19-5, a single part is driving too many inputs on SIZE-replicated parts.

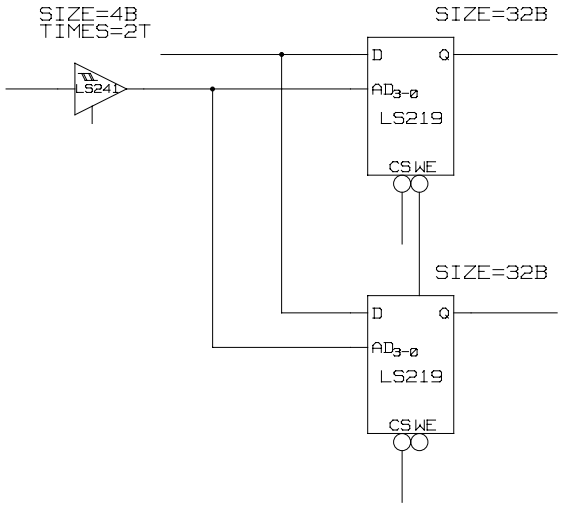
Figure 19-5 Structured Design with the SIZE Property



In this design, the 4-bit 3-state buffer drives 64 bits of memory. Four sections of LS241 do not have the drive capability to handle 16 memory packages; Packager-XL would report a loading error.

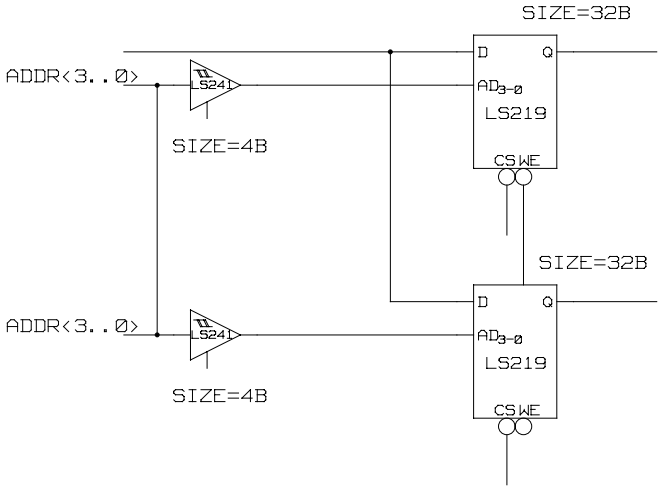
The TIMES property is used to correct loading violations in structured designs, as illustrated in Figure 19-6.

Figure 19-6 Using the TIMES Property



In this example, the TIMES property informs the system that two instances of a 4-bit, 3-state buffer are needed. The system checks the loading and balances the load between all the parts being driven. Using the TIMES property in this design is equivalent to adding another part and more interconnections, as illustrated in Figure 19-7 on page 514.

Figure 19-7 Manually Balancing Loads



Using the TIMES property eliminates the need to manually balance the load and enter more data.

The Standard Library

Cadence provides a Concept HDL library of standard parts that allow you to define and manipulate signals in a structured design. The Standard library is automatically associated with your search list of libraries so that you can conveniently use these parts in your designs. Although the bodies in the Standard library can be used for any of the design techniques, many of them are created especially for structured designs.

The library contains merge bodies for merging signals, tap bodies for tapping bits from buses, and several other special parts.

Benefits of Structured Designs

Using a structured design technique has these advantages:

- Creating structured designs can dramatically decrease the design cycle time.

  The amount of data entered into Concept HDL is reduced, resulting in faster schematic entry. Also, the analysis tools run more efficiently on structured designs because they can process multiple bits in parallel.

- Errors in design entry are minimized because of the reduced number of parts and simplified interconnections.

- The resulting print is less cluttered, easier to read, and easier to understand.

Considerations of Structured Designs

Packager-XL produces an easy-to-read cross-reference list for the logical-to-physical mapping of the design data. These lists are used with the structured print set for design troubleshooting. Members of the design team responsible for troubleshooting the structured design must be educated on how to read structured print sets and how to reference the physical information.

Hierarchical Designs

The hierarchical design technique is an efficient approach to developing complex designs that can be organized into modules. This method is useful for designs that re-use many of the same circuit functions and for isolating portions of the design for teamwork assignments.

A hierarchical design results in print sets that are easy to read and produces modules that can be effectively debugged. Hierarchical designs, like structured designs, reduce the

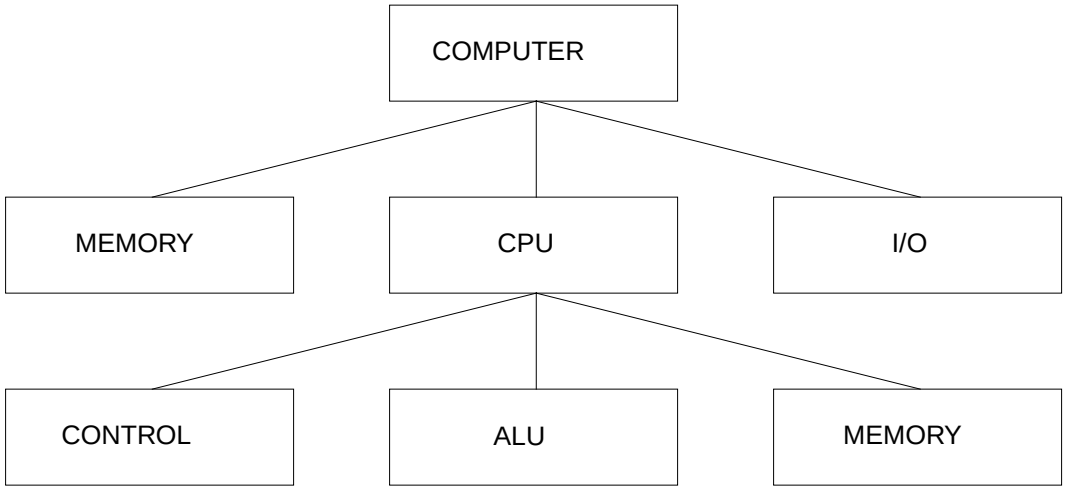
amount of data entry and interconnections required by the design, thereby reducing the chance for error. Also, all the design tools can be used to analyze partial designs (modules).

Creating a Hierarchical Design

Creating a hierarchical design is a natural extension of the entire design process. If the design to be implemented is a computer, the design begins by planning the constituent parts of the computer.

The computer can be divided into the CPU, MEMORY, and I/O modules. The CPU module can be further divided into the ALU, MEMORY, and CONTROL modules. This represents three levels of hierarchy in the design. There are no limits to the number of levels you can include in a hierarchical design. Figure 19-8 on page 516 shows the hierarchical levels of the computer.

Figure 19-8 Levels of Hierarchy



After you plan the modules of the design, you implement the design using the following basic procedure:

1. Create a schematic drawing that represents a functional portion (module) of your design (for example, counter, register file, memory unit, or control blocks of circuitry).

   You can start at the most detailed level of the design hierarchy.

2. Test that drawing, processing it with other system design tools to check its timing and logic functions.

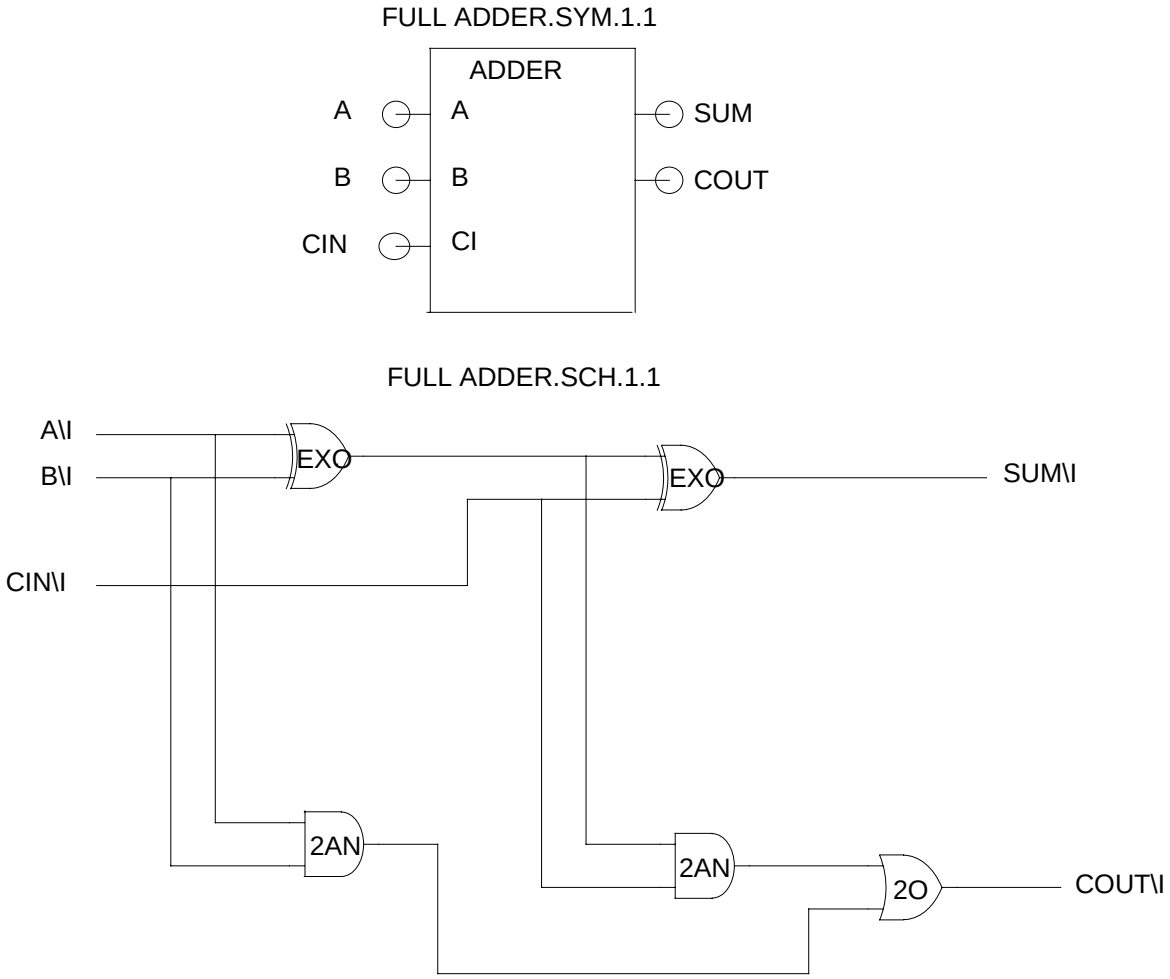
   You can efficiently debug each module of the design as you work.

3. Create a symbol drawing to represent the design module.
4. Create a new schematic drawing and add the required number of symbol representations to it, building a circuit using the modules.

   You have added a symbol that represents the functional module you created in Step 1. The symbol drawing acts as a pointer to the schematic definition of the circuit.
5. Continue to create the corresponding schematic/symbol representations for each of the defined modules in the design, working up the levels of hierarchy.

Figure 19-9 on page 517 illustrates the schematic and symbol drawings defined for use in a hierarchical design. Instead of having to wire together the gates of the Full Adder circuit whenever it is needed, you add the `Full Adder.body` drawing in its place.

Figure 19-9 Full Adder Logic and Symbol Drawings



Every level of hierarchy (except the lowest level) is made up of a schematic drawing and symbol drawing pair. The schematic drawing defines the functional circuitry for the design module. The symbol drawing is the picture or symbol that represents the logic function. The symbol points to the functional representation, but does not take up as much space in higher levels of the hierarchy. The result is a well organized and understandable design print set.

Creating Symbols

When you create a hierarchical design, you draw simple blocks (also called symbols) to represent the specific logic for each element of the design. Concept HDL provides you with the tools for drawing bodies and establishing the relationships between the symbol drawings and the logic drawings they represent.

The pins on the bodies that correspond to signals in the logic drawing must have the same name. Additionally, these signals in the schematic drawing are given the interface signal property (\I). This signal property is used to indicate an interface signal from a higher level drawing.

There are two ways to make a schematic drawing in Concept HDL.

- Use *Tools > Generate View* to automatically create a symbol drawing either from an existing schematic drawing or from a list of pins.
- Use *File > New* to create the symbol by drawing.

  You can also use *Tools > Generate View* and then edit the symbol drawing to move the pins or change the shape.

Editing Symbols

To create a symbol drawing by editing it,

1. Use *File > Open* to edit a symbol drawing.

   The *View Open* dialog box is displayed.
2. Specify the name of the *Cell*.
3. Select the *View* as `Symbol`.

   A grid is displayed with a cross to mark the origin of the symbol. The default grid for symbol drawings is 0.05 inches, and every second grid point is displayed. This default grid is twice as fine as the default grid for logic drawings.

To change the grid for symbol drawings, use the `set` command option `default_symbol_grid`. To change the grid for schematic drawings use the set command option `default_grid`. Both options require a numeric grid size argument.

4. Use *Edit > Move* to move the name away from the origin.

   You can also move the origin, but be very careful if you do so. Do not place the origin at a connection point (pin end) for the symbol.

5. Use *Wire > Draw* to draw the outline of the symbol around the origin symbol.

   The grid is used as a guideline for the appropriate size and shape of the symbol.

6. Add wire stubs for the pins.

   Make them 0.1 inch (one grid segment) long and place them on visible grid lines, so that the symbol can be correctly wired on schematic drawings.

   Be sure to place pins only at visible grid points on the symbol drawing. This guarantees that all of the symbol pins will be on-grid when the symbol is used in a logic drawing. Use the unmarked grid points on the symbol drawing only for placing notes and properties.

   See "Defining Low-Asserted Pins" on page 521 for information about defining low-asserted pins.

7. Use *Wire > Dot* to place a dot at the end of each pin.

   Place dots on displayed grid intersection points. Use the right mouse button to ensure that the dot is properly placed at the end of the wire.

8. Use *Wire > Signal Name* to add pin names (corresponding to the signal names in the related logic drawing) to each pin.

   In a symbol drawing, the SIG_NAME properties added by *Wire > Signal Name* are understood as PIN_NAME properties. They can only be attached to pin connections. The name must match the corresponding name in the logic drawing, except for the omission of the interface property (\I). Use *Display > Attachments* to ensure that all pin names are attached properly.

9. Use *Text > Note* to place labels within the symbol drawing.

   This makes the purpose of the symbol and each pin clear.

10. Mark the clock signal with a wedge. Choose *Wire > Draw* and press the middle button to draw diagonal lines.

11. Use *File > Save* to save the symbol drawing.

The pinnames Command

When you create a hierarchical schematic drawing and symbol drawing pair, you can use the PIN NAMES symbol in the `standard` library to transfer the PIN_NAME properties from the symbol drawing to the associated schematic drawing. When you add the PIN NAMES symbol to a schematic drawing, all the pin names in a symbol drawing of the same name are attached to the PIN NAMES symbol. You can reattach the names to appropriate signals in the schematic drawing. This eliminates the need for retyping signal names and reduces mislabeled signal names or missing interface scope (`\I`) signal properties.

The `pinnames` console window command adds a PIN NAMES symbol to a schematic drawing and attaches the pin names to the symbol. To do this,

1. Use *File > Open* to edit a symbol drawing.
2. Add pin names to the symbol using *Wire > Signal Name*.

   When used in a symbol drawing, *Wire > Signal Name* attaches a PIN_NAME property to the specified pin.
3. Save the symbol drawing using *File > Save*.
4. Create a schematic drawing by the same name as the symbol drawing.

   For example, if the symbol drawing is `CLOCK.SYM.1.1`, type the following in the Concept HDL console window:

   ```
   edit clock
   ```

   The `CLOCK.SCH.1.1` drawing will contain the logic that the symbol represents. Place all the required parts and attach wires as required.
5. Type `pinnames` in the Concept HDL console window.
6. Click to add the PIN NAMES symbol to the `CLOCK.SCH.1.1` drawing.

   Each pin name defined on the symbol drawing `CLOCK.SYM.1.1` appears in the schematic drawing attached to the PIN NAMES symbol. A `\I` suffix (scope = interface) is also suffixed to each signal name. When transferred to the schematic drawing, each pin name is identified with a `SIG_NAME` property.

   Do one of the following if you want to view the signal and property names:

   - Place the cursor on a pin name attached to the PIN NAMES symbol
   - Choose *Text > Attributes* and click on the PIN NAMES symbol to view the signal and property names in the *Attributes* dialog box.

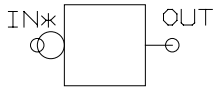
   Note: If you choose *Component > Add* and add the PIN NAMES symbol on the schematic drawing, the pin names on the symbol drawing will not get automatically

attached to the PIN NAMES symbol. Run the check console window command. The pin names on the symbol drawing are now attached to the PIN NAMES symbol on the schematic drawing.

7. Move the PIN NAMES symbol to an unused area of the schematic drawing.
8. Choose *Text > Reattach* to reattach the individual signal names from the PIN NAMES symbol to the appropriate signals on the schematic drawing.
9. Choose *Display > Attachments* to ensure that the signal names have been reattached to the appropriate signals.
10. For drawing clarity, choose *Edit > Move* to relocate the signal names near the associated signals.
11. Delete the PIN NAMES symbol.

Defining Low-Asserted Pins

Use a circle instead of a wire to represent a low-asserted (bubbled) pin. You can use either *Edit > Circle* or *Edit > Arc* to add circles. The circle should be 0.1 inch in diameter. A dot is placed on the appropriate grid intersection point on the circumference of the circle to mark the connection point. The signal name should also be low-asserted (*).



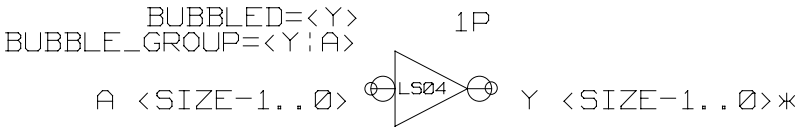
To define pins that can be either bubbled or unbubbled,

➤ Draw a symbol and represent the pins with both wires and circles.

There must be a line that extends across the diameter of the circle so that both representations are available. Be sure to place a dot at the connection point. You also attach the BUBBLE property to the origin of the symbol to define which pins are bubbled when the part is added to a drawing.

You can also define groups of pins that automatically change state when one of the pins in the group is bubbled. These are called bubble groups.

Figure 19-10 Using Bubbleable Pins



When you add the symbol to a drawing, use *Component > Bubble Pins* to toggle the pin from bubbled to unbubbled.

Benefits of Hierarchical Designs

The benefits of creating hierarchical designs are similar to those of structured designs:

- Creating hierarchical designs can dramatically decrease the design cycle.

  Since symbol drawings act as pointers to schematic drawings, a large amount of data entry need not be repeated.

- The functional schematic drawing that a symbol represents can be compiled once and linked to all locations where the symbol is used.

- The number of entry errors is minimized because the amount of schematic entry is reduced.

- Each module can be fully tested before it is incorporated into higher levels of the design because functional modules are created when defining a hierarchical design.

  Testing can be performed incrementally rather than at the end of the design process.

- Hierarchical designs result in designs that are well organized and easy to read and understand.

Comparing Design Techniques

The design methodologies discussed in this chapter (flat, structured, and hierarchical) are all appropriate for solving design problems. You must weigh the benefits and considerations of each technique before deciding which method to use.

There is no restriction against combining these methods in design drawings. Hierarchical and structured design techniques are often used together to provide maximum flexibility and efficiency for the design engineer.

Flat Designs

Best suited for	Small designs
	Designs that do not re-use modules
	Designs that do not use buses
Benefits	Short learning curve
Considerations	Long design cycle time
	Cluttered print sets
Structured Designs	
Best suited for	Designs that use sophisticated bus structures
Benefits	Shortened design cycle time
	Fewer errors during data entry
	Less cluttered print sets
	Print sets organized in the logical flow of the design
	Cross-reference listings
Considerations	Additional training required for design troubleshooters
Hierarchical Designs	
Best suited for	Designs that re-use modules
	Large designs that can be organized into separate components
Benefits	Shortened design cycle
	Fewer data entry errors
	Easy-to-read print sets
	Cross-reference listings
	Effective debugging capability
	Print sets organized in logical (top-down) flow of design
Considerations	Additional training required for design troubleshooters

A

Console Command Reference

This section describes the commands that you can enter in the Concept HDL console window. The syntax, abbreviation, description, and related commands for the following console commands are described below:

Add

Arc

Assign

Attribute

Auto

Backannotate

Badd

Bpadd

Bindview

Bpdelete

Bpmove

Bprename

Brename

Broute

Browse

Bstretch

Bubble

Busname

Bustap

Bwire

Change

Check

Circle

Copy

Dehighlight

Delete

Diagram

Directory

Display

Dot

Echo

Edit

Error

Exclude

Exit

Filenote

Find

Get

Gotosheet

Grid

Group

Hardcopy

Highlight

IGnore

Include

Library

Loadstrokes

Mirror

Modify

Move

Next

Note

Paint

Pause

Pinnames

Pinswap

Plot

PPTAdd

PPTDelete

PPTEcho

Property

Quit

Reattach

Recover

Redo

Remove

Replace

Return

Rotate

Route

s2l

Scale

Script

Searchstack

Section

Select

Set

Show

Signame

Smash

Spin

Split

Strokefile

Swap

System

Tap

Textsize

Undo

Unhighlight

Unix

Updatesheetvars

Use

Vectorize

Version

Vpadd

Vpdelete

Window

Wire

Write

Zoom

Add

Syntax

```
ADd [component_name][.view][.version] [point] [point...] | <cr>
```

Description

This {command} adds a specified component to a drawing. Component_name is the name of the component to be added. View is the symbol view. The version is 1 by default, but any existing version of a component can be added.

To add a component to a schematic, type ADD <*component_name*>. A copy of the component is attached to the cursor. Press the right mouse button and select *Version* to cycle through different versions of the component. Move the component to the required position in the drawing and click. To add another copy of the component, click, then position the copy as required.

ADD <cr> accesses the Component Browser dialog box. You can use this dialog box to add components to the drawing. The Component Browser dialog box can be turned off by typing in SET ADDFORM OFF.

Use the REPLACE command to substitute one component for another.

Related Commands

Rotate
Replace
PPTAdd
Version

Arc

Syntax

```
ARc point1 point2 { point3 | ; }
```

Description

This command creates arcs, usually on symbols. The two points define the ends of the arc. The curvature of the arc is controlled dynamically by dragging the mouse after you place the second point.

Click to place the arc at the nearest screen pixel. Click the center button to place the arc at the nearest grid intersection (useful when building accurate semicircles or matching mirrored arcs).

Typing a semi-colon after placing the second point will create a circle.

Related Commands

Circle

Assign

Syntax

```
Assign function_key "quoted-string"
```

Description

This command assigns an editor command or operation to a programmable function key. You can press the specified key instead of typing the text. This saves time when a command is used often or requires several variables and options on the command line.

To assign a string to a key, enter the key name or press the function key and then type in the command text to be assigned to the key. Enclose the command and its arguments in quotation marks. They can be uppercase or lowercase. Note that the shift and control keys can also be used, thus allowing up to three different assignments to each function key.

For example,

```
ASSIGN F2 "zoom fit"
ASSIGN <shift>F2 "zoom in"
ASSIGN <cntrl>F2 "zoom out"
```

Function key names correspond as closely as possible with the text printed on the keyboard. The function keys for the various systems are:

IBM: F6-F10, F12, 4-9, page up/down, the directional arrows

Sun: F1-F9, R4-R12

SHOW KEYS lists the current function key assignments. ECHO <key press> displays the key name or the assignment.

Attribute

Syntax

```
Attribute point
```

Description

This command accesses the Attribute Form for adding, modifying, or deleting properties on a drawing. To use the command, type ATTRIBUTE, point to the desired object, and click the left mouse button. The editor brings up a form containing all the properties attached to the selected object.

Auto

Syntax

```
AUto {PAth | Dots | Undot | PRoperty group_name prop_name prop_value . . }
```

Description

This command performs the global addition or deletion of certain objects to or from a drawing.

- PATH automatically assigns unique PATH numbers to bodies on a drawing that do not already have a PATH property. (Some special editor bodies and bodies labeled with a COMMENT property are not assigned PATH properties.) The path property is in the form 'PATH = In', where n is a unique integer.
- DOTS places a dot at each wire connection on the current drawing. Open dots are the default value. To specify filled dots, type SET DOTS_FILLED before entering AUTO DOTS.
- UNDOT removes all dots from the drawing except those at the intersection of four wires.
- PROPERTY will add user properties onto groups of components. GROUP_NAME is either a single letter identifying a group or a mouse click specifying the nearest group. Any number of property name-value pairs may be specified after the group, and the names and values may be separated by spaces, an equal sign, or a new line. Using this option will cause the entered properties to be automatically annotated onto the components within the group.

See also the SET DOTS_FILLED and SET DOTS_OPEN commands.

Auto Commands

Using the auto dot command

You can use the auto dot command to place dots on a complex circuit. The auto dot option with the `set` command automatically places dots on a drawing as you are creating it (`set autodot on`). Automatic dotting places dots at all intersections with an odd number of wires.

1. When `set autodot` is `off`, type

   ```
   show connections
   ```

   This command places asterisks temporarily on the drawing to highlight each connection point.

2. Check the drawing to make sure that no connections have been made by mistake.
3. Use the refresh command to remove the asterisks from the screen.
4. Type

   ```
   auto dot
   ```

   All the junctions are automatically dotted.

Using the auto undot command

To remove the dots of the same size as the present dot size settings at intersections in a drawing type

```
auto undot
```

Using the auto allundot command

To remove all the dots at the intersections in a drawing type

```
auto allundot
```

Changing Dot Size

- If you change the size of a dot, the change is not reflected on filled dots in the schematic, however, when you plot the schematic, the change in size gets reflected.
- Change in size is also not reflected in dots that existed previously, it appears only in the dots that you add after changing the size. So, if you want to make all the dots of the same size, first use `auto undot` to remove the dots within the size specified in logic dot radius, or `auto allundot` to remove all dots regardless of size. Then add the dots again using `auto dot`.

Using the auto path command

If the SET command option AUTOPATH is on, the PATH property is automatically added to a part when it is added to a drawing. If `set autopath` is off, you can use the `auto path` command to assign PATH properties to symbols on a drawing that do not already have a PATH property.

Using the auto property command

To attach properties to symbols in a group, type

```
auto property <group_name> <property_name> <property_value>
```

Using the autoroute command

The `set` command option `autoroute` on activates automatic routing after moving an object in the direct mode. The set autoroute option can be turned on or off.

Backannotate

Syntax

```
Backannotate {annotation file | <cr>}
```

Description

This command annotates designs with physical information from the Packager. The editor reads the specified schematic annotation file produced by the Packager. The file includes physical information such as location designators, pin numbers, physical net names on the design, and user-defined properties, if any.

The annotated properties added by the editor are soft properties. Soft property names begin with a dollar sign (for example, $LOCATION) and are not written into the connectivity file. This allows Packager to reassign physical information each time the design is repackaged.

You can move and delete soft properties, or you can change a soft property into a hard property by using the PROPERTY command and adding a property with the same property name without the dollar sign.

To generate a backannotation file, use the following directive when running the Packager:

```
output backannotation;
```

By default, this directive is set.

To process the backannotation file generated by the Packager (`pstback.dat`), type either of the commands BACKANNOTATE PSTBACK.DAT or BACKANNOTATE <cr>. BACKANNOTATE <cr> brings up the File Browser form. The user can then select the name of the backannotation file from the form. The editor reads the file, edits each named drawing in turn, adds the appropriate physical information, and writes the drawing. Backannotation aborts if any errors are detected during the process.

Caution

Do not run backannotation if any other user who has write permissions is working on the design. Running backannotation when another user is working on the design results in incomplete backannotation.

Limitations of Backannotation

1. Sizeable parts having property `SIZE > 1` are not backannotated.
2. Net properties are backannotated only if they have existing placeholders.
3. Properties on buses are not backannotated unless they exist on individual bits of the bus.

See also the SET command options that control property visibility and pin number placement.

Badd

Syntax

```
BADd { [block_name] point1 point2 } . . .
```

Description

This command creates and add blocks. The block is a rectangle between point1 and point2. If you omit block_name, Concept HDL automatically names the block (the name will be "BLOCK" followed by an integer). If you do enter a block_name, and a symbol for that part already exists, you will be thrown into the add command.

Related Commands

Add
Bpadd
Bpdelete
Bprename
Bpmove
Brename
Broute
Bstretch
Bwire

Bpadd

Syntax

```
BPAdd {pin_name point}...
```

Description

Add or rename pins on blocks. bpadd adds pins to blocks with a user-specified name. It is useful if you want to define the interface pins of a block before you connect up your blocks. pin_name is the name of the pin that will be created. Point is where the pin is created.

Related Commands

Badd
Bpdelete
Bprename
Bpmove
Brename
Broute
Bstretch
Bwire

Bindview

Syntax

```
Bindview point1 point2
```

Description

This command makes a drawing that is visible in one viewport visible in another viewport also. Any changes you make to one copy of the drawing also appear on the other copy.

Point1 selects the drawing to make visible. Point2 specifies the new viewport where the drawing should appear.

When you use BINDVIEW, the drawing name is not added to the drawing stack in the second viewport. If you issue a command, such as SHOW or RETURN, the bound drawing does not appear in the list. When you EDIT or RETURN to a drawing in the bound viewport, the binding is removed. Use BINDVIEW again to re-bind the drawing.

Bpdelete

Syntax

```
BPDelete {point}...
```

Description

This command deletes pins on blocks. You can choose the pins to be deleted by pointing to them with the mouse.

Related Commands

Badd
Bpadd
Bprename
Bpmove
Brename
Broute
Bstretch
Bwire

Bpmove

Syntax

```
BPMove {point1 point2}...
```

Description

This command moves a pin from point1 on a block to point2.

Note: If you try to undo this operation, some hanging properties may remain on the schematic. This is because the `bpmove` command actually moves a pin and its properties on the symbol while the `undo` command operates only on the schematic. To remove these hanging properties, perform the following steps:

1. `set sticky_on`
2. `get`
3. `set sticky_off`

Related Commands

Badd
Bpadd
Bpdelete
Bprename

Brename
Broute
Bstretch
Bwire

Bprename

Syntax

```
BPRename {pin_name point}...
```

Description

This command renames pins on blocks. In this command, pin_name is the new name of the pin and point is location of the pin.

Related Commands

Badd
Bpadd
Bpdelete
Bpmove
Brename
Broute
Bstretch
Bwire

Brename

Syntax

```
BREname {[block_name] point1}...
```

Description

This command renames blocks. This is useful if you allowed the system to generate default names for your blocks and want to rename them. It DOES NOT remove old blocks from the disk. It does, however, remove the old block from the memory if you are not actually viewing it. So if the old block had not been saved to disk, it will have been removed. If the old block had been saved to the disk, you can remove it from disk with the REMOVE command.

This command is also useful if you want to create a block similar to one you already have. You can copy the block (with the COPY command) and rename the copy of the block. The original block will be unaffected.

Related Commands

Badd
Bpadd
Bpdelete
Bpmove
Bprename
Broute
Bstretch
Bwire
Copy
Remove

Broute

Syntax

```
BROUte {[signal_name] point1 point2}...
```

Description

This command routes a signal named signal_name between point1 and point2. If the signal_name is a bus name, create a heavy wire.

Related Commands

Badd
Bpadd
Bpdelete
Bpmove
Bprename
Brename
Bstretch
Bwire
Route
Wire

Browse

Syntax

```
Browse {<libname> | <cr>}
```

Description

This command accesses the COMPONENT BROWSER dialog box. It lets you scan through all libraries and active directories. You can also access the COMPONENT BROWSER with the ADD command.

BROWSE <*libname*> sets the library field to the given library name and resets the selection list to that library.

Bstretch

Syntax

```
BStretch {point1 point2} . . .
```

Description

This command resizes blocks. Select the side or corner of the block you want to stretch with point1. Then select where it stretches to with point2. Pins will move with the side they are attached to. Pins pointing to the left or right will only move horizontally. Pins pointing to the top or bottom only move vertically. You may not shrink a block so much that its pins or its origin fall outside it.

Wires attached to the instance of the block being stretched will be attached to new pin locations. However, they are not re-routed. If a new wire stub is not straight, use the SPLIT and DELETE commands to straighten it.

Related Commands

Badd
Bpadd
Bpdelete
Bpmove
Bprename
Brename

Broute
Bwire
Delete
Split

Bubble

Syntax

```
BUBble point...
```

Description

This command toggles the state of a pin between bubbled and unbubbled. Bodies must be defined with bubbleable pins to permit this conversion. If the pins are established as part of a bubble group, the BUBBLE command can be used to convert the symbol from one form to another.

For example, a NOT symbol is defined with both the BUBBLED and BUBBLE_GROUP properties attached:

BUBBLED=(B)

BUBBLE_GROUP=(A|B)

Because BUBBLED is equal to (B), pin 'B' is bubbled when the part is initially added to a drawing. If you type BUBBLE and point to either pin A or B, the attached BUBBLE_GROUP property specifies that pin A is now the bubbled pin and pin B the unbubbled pin.

Busname

Syntax

```
BUSName bus_name point point
```

Description

This command places single-bit vectored signal and pin names on a drawing. For example, the bus name A<7..0:2> results in the signal names A<7>, A<5>, A<3>, and A<1>.

To use the BUSNAME command, first place separate wires or pins (use the COPY command). Next, enter the bus name in Concept HDL signal syntax (for example, DATA<15..0>\I). Then, select two points. Concept HDL will find all the wires you have crossed and add the properties to those wires.

For examples, if you issue the following command:

```
busname a<20..0:2>\I
```

then pick two points, such that you have crossed over three wires, the crossed wires will be named as `a<20>\I`, `a<18>\I` and `a<16>\I` respectively, then you can select another two points and the wires crossed will be named `a<14>\I` and `a<14>\I` and so on.

If the first thing you do after entering the BUSNAME command is to enter a point, the editor uses the most recently entered signal name. This is useful if you place the first two signal names incorrectly. If the attachment lines show that the names are not connected to the appropriate signals, use UNDO to delete the incorrect attachments, then type BUSNAME and click on the drawing area. You can then reposition the first two names.

Related Commands

Signame
Bustap

Bustap

Syntax

```
BUSTap bus_tap_value point point...
```

Description

This command fills in the value of the BN (bit number) properties on bus taps in drawings.

The bus_tap_value should be in the Concept HDL signal syntax with the first number as the start tap value, the second number as the end tap value, and the third number the increment. For example, a bus_tap_value of 7..0:2 results in BN values of <7>, <5>, <3>, and <1>, with <7> being placed on the tap closest to the first point, <1> on the tap closest to the second point, and <5> and <3> on the taps in-between.

The third number is optional. If no third number is specified, for example, 7..0, an increment of 1 is assumed. The second number is also optional, and if not specified, for example `7', all taps pointed to will have a BN value of the first number.

Bwire

Syntax

```
Bwire {[signal_name] point  point }  . . .
```

Description

This command adds wires to a drawing. The wire begins at the first point specified and runs to the second. Specify additional points to draw a wire with more segments. To snap the wire to the nearest vertex, right click. To end a wire at a pin, dot, or other wire, press the left button. To end a wire in a free space, press the left button twice at the final point.

See also the WIRE, BROUTE, ROUTE commands, and the SET and SHOW commands to change default wiring behavior.

Related Commands

Broute
Route
Set
Show
Wire

Change

Syntax

```
Change {group_name | point}...
```

Description

This command modifies selected lines of text in place. The selected text items will be highlighted, and the cursor will be placed on the first text item. For notes, the cursor is placed before the first character. For properties, if both the name and the value is visible, the cursor is placed after the = character. If only the name or the value is visible, the cursor is placed at the beginning. After changing one line of text, the user can move over to the next text item by typing <cr>. The changes made to the line of text are then committed. The changes made to each line of text can be undone using the UNDO command.

The set option INLINE_TEXTEDIT can be used to toggle between two modes of using the change command. When the option is ON (it is, by default), the user can edit lines of text, in place.

The following commands are useful for editing text:

<BackSpace>	Deletes the previous character
<Delete>	Deletes the previous character
Ctl<k>	Deletes to the end-of-line
Ctl<a>	Moves the cursor to the beginning of the line
Ctl<d>	Deletes the next character
Ctl<e>	Brings up the selected text items in an editor
<-	Moves backward
->	Moves forward

Check

Syntax

```
CHEck <cr>
```

Description

This command checks for connectivity problems and general errors on the current drawing. An option exists to allow the user to turn the check on or off. Concept HDL performs the following checks on the schematic:

- Duplicate components in the same location

  set option: CHECK_PARts_at_same_loc <ON/OFF>

  To correct the error, use the `split` console command and separate overlapping elements. For more information on the `split` console command, see Split on page 596.

- Pins attached to more than two wire segments (this may not be an error, but is an error if a wire inadvertently shorts the pins on a device)

  set option: CHECK_TWo_wires_at_pins <ON/OFF>

- Wires connected to only one pin and not named (NC wires)

set option: CHECK_Unconn_wires <ON/OFF>

- Nets that are named but not connected to any pins

  set option: CHECK_SIGNAMES <ON/OFF>

- Wires that come close to but do not contact pins

  set option: CHECK_PIN_near_wire_endpt <ON/OFF>

- Missing TITLE and/or ABBREV properties

  set option: CHECK_TItle_abbrev <ON/OFF>

- Bodies that are placeholders

  set option: CHECK_Body_place_holders <ON/OFF>

- Pins located at the origin (0,0) in BODY drawings

  set option: CHECK_PINS_at_origin <ON/OFF>

- Multiple dots at the same location

  set option: CHECK_Arcs_at_same_loc <ON/OFF>

  To correct the error, use the `split` console command and separate overlapping elements. For more information on the `split` console command, see Split on page 596.

- Hard properties with the? value (placeholders)

  set option: CHECK_PRop_place_holders <ON/OFF>

- Wires connecting the pins of a two-pin body

  set option: CHECK_SHorted_pin <ON/OFF>

- Wire segments hidden by parts of a body

  set option: CHECK_Hidden_wires <ON/OFF>

  To correct the error, use the `split` command and separate overlapping elements.

- Pin properties which are no longer attached to pins.

  set option: CHECK_Missing_pins <ON/OFF>

  To correct the error, reattach properties to new pins, delete properties or replace part.

- Inconsistent section properties

  set option: CHECK_PACk_sec_type_props <ON/OFF>

To correct the error, resection the part.

- Signame properties defined within a symbol (body).

  set option: CHECK_SIGNAME_in_body <ON/OFF>

  To correct the error, remove signame properties from the body files.
- Duplicate PATH properties
- Wires overlapping a body
- Check for legal HDL net names (hdl_direct on only)

  set option: CHECK_Net_names_hdl_ok <ON/OFF>
- Check for legal HDL port names (hdl_direct on only)

  set option: CHECK_POrt_names_hdl_ok <ON/OFF>
- Check for legal HDL symbol names (hdl_direct on only)

  set option: CHECK_SYmbol_names_hdl_ok <ON/OFF>
- Run checks automatically when writing drawings.

  set option: CHECK_On_write <ON/OFF>

CHECK lists each detected error. After you run the CHECK command, you can use the ERROR command to locate each error on the drawing.

Related Commands

Error

Circle

Syntax

```
Circle point point
```

Description

This command adds circles to a drawing. To place a circle on the drawing, enter the CIRCLE command and select a point as the center of the circle. To size the circle dynamically, drag the mouse and then click again to place the circle on the drawing.

Circles and arcs are rarely necessary on logic designs but are commonly used for creating symbol drawings.

Related Commands

Arc

Copy

Syntax

```
COPy {[count][REPEAT][ALL] source_point destination_point | [count][REPEAT][ALL]
group_name destination_point | property_point destination_point attach_point }
```

Description

This command copies objects, properties, and groups in the current drawing or between viewports.

COUNT indicates the number of copies to place on the drawing. To make multiple copies, type COPY and enter a number to specify the number of copies to make. Move the cursor to the object or group to be copied and click to select an object or the center button to select a group. Click to place the copies at grid points or the right button to place copies at the vertex nearest the cursor. After you place the first copy, the remaining copies are automatically added to the drawing. The second copy is offset from the first copy by the same distance as the first copy is from the original. You can use this feature to copy single items and groups.

SOURCE_POINT is the object to copy, PROPERTY_POINT is the property to copy, and DESTINATION_POINT is the position point for the new copy. When you copy a property, ATTACH_POINT attaches the property to an object (symbol, pin, rewire).

To copy an object (such as a symbol or a wire), type COPY, position the cursor on the object, and press the appropriate button. The left button picks up a copy of the object at the grid point nearest the cursor. The right button picks up a copy of the object at the vertex nearest the cursor. (The vertex of the copy snaps to the cursor.) This is useful for copying component bodies and wires. Click to place the copy on the grid point nearest the cursor or the right button to attach the copy to the nearest vertex (useful for attaching copies of wires at new locations).

To copy a group, use the GROUP or SELECT command to define a group, then type COPY. Move the cursor to the group to be copied and click the center button to select the nearest

group. You can also type the single-letter GROUP_NAME and press <cr>. Click to place the copy.

To copy properties, type COPY, and click to select the property to copy. Move the cursor to the location for the copy and click. A flexible line is drawn from the property to the cursor. Move the cursor to the object where the property is to be attached and click. You can attach the property to a part, wire, pin, or signal name.

You cannot copy default symbol properties, soft properties, PIN_NUMBER properties, or properties generated by the SECTION, PINSWAP, and BACKANNOTATE commands. User-added properties are included in copies of parts. Signal names are not copied. Wire properties are not included when you copy a wire. If a default symbol property on a symbol was changed, the copy of the symbol contains the changed value.

There are two options to the COPY command. COPY ALL will copy section properties, soft properties, pin properties, wire properties, and properties attached to other properties. This option is especially useful if you are copying a section of logic from one drawing to another. If you want to place a copy of something in several unrelated places, try using the COPY REPEAT option. This option causes the copy command to reselect the objects you originally selected after you have placed a given instance. The REPEAT and ALL options may be used together.

Groups of properties are not copied. When applicable, properties attached to objects are copied with the group.

Dehighlight

Syntax

```
DEHighlight [ Net | PArt | PIn | Any ] pt
DEHighlight [ Net | PArt | PIn | Any ] object_name
```

Description

Net, Part and Pin are used to specify the object type you want to unhighlight. You may use Any if you want all selected objects to be unhighlighted. This is also the default argument if you do not specify any object type

.

pt	To pick a component to be unhighlighted, point to the object and press the left button. Select a group by pointing to the required group and pressing the middle button.

object_name	If you would prefer typing in the name of the object, you may do so by using the second version of this command.

For nets, the object name is the signal name of the net, for example, FOO.

For parts, the object name is the value of the PATH property attached to the component, for example, 7P.

For pins, the object name is the value of the PATH property attached to that component to which the pin belongs, followed by a period ("."), followed by the name of the pin, for example, 7P.A<SIZE-1..0>.

Wildcards such as * and ? may be used in specifying the object name,

For example, FOO*, 7*P, 7P.A*.

The Dehighlight command is used to unhighlight an already highlighted object throughout the system.

Related Commands

Highlight
Unhighlight

Delete

Syntax

```
DELete {point | group_name}...
```

Description

This command removes objects from a drawing. To delete an object, point to any part of the object and press the left button. To delete a group, use the center button or type in the single-letter group_name. DELETE removes the object or group nearest to the cursor.

You cannot delete default properties on bodies and pin number properties generated by the PINSWAP command.

The UNDO command lets you retrieve groups or objects deleted by mistake.

Related Commands

Undo

Diagram

Syntax

```
DIAgram [<library>]cell[.type][.version][.page]
```

Description

This command works like the *File > Save As* menu option in Concept HDL. You can use an existing drawing as a pattern for a new drawing or save a copy of a drawing by a different name before making changes to it.

- <LIBRARY> is the name of the library where the drawing resides. The library name must be enclosed in angle brackets. If no library is specified, the current library is the default.
- CELL is the new name of the drawing. The current drawing name is taken by default. For example, if the Concept HDL title bar displays ATM.SCH.1.2, the current drawing name is ATM.
- TYPE is the drawing type. The drawing type can be SCH (schematic), SYM (symbol). If no drawing type is specified, the current drawing type is used. For example, if the Concept HDL title bar displays ATM.SCH.1.2, the current drawing type is SCH.
- VERSION is the version number of the drawing type. For example, if the Concept HDL title bar displays ATM.SCH.1.2, the version number of the drawing type SCH is 1. If no version number is specified, the default value 1 is used.
- PAGE is the page number for the drawing. For example, if the Concept HDL title bar displays ATM.SCH.1.2, the page number of the current drawing is 2. If no page number is specified, the default value 1 is used.

To rename a drawing, edit the drawing to be changed, type DIAGRAM and the new name of the drawing. Type WRITE to save a copy of the drawing by its new name.

For example, to use the drawing SHIFTER.SCH.1.1 as a pattern for a new drawing named NEWSHIFTER.SCH.1.1, use the commands:

```
EDIT SHIFTER.SCH.1.1
DIAGRAM NEWSHIFTER
WRITE
```

The NEWSHIFTER.SCH.1.1 drawing is saved to disk.

Directory

Syntax

```
DIRectory {[<directory>][name][.[type][.[vers][.[page]]]] |<cr>}
```

Description

This command lists the names and contents of directories in the current directory list in the order that the directories are searched with the current working directory displayed first.

DIRECTORY <cr> accesses the DIRECTORY BROWSER form. The DIRECTORY BROWSER form shows the current library or directory. Any items listed in the left section of the form are from the current directory or library. To list the contents of a different directory or library, select the Current Dir/Lib field and type the directory or library name. The directory form can be turned off by typing in SET DIRFORM OFF.

<DIRECTORY> is the directory name whose contents you want to list. The name must be enclosed in angle brackets. If no directory is specified, the current directory is taken by default. NAME is the drawing to be listed. Unless you specify the drawing type, the version number, and the page number, the DIRECTORY command displays only the drawing name. You can also list drawings by type, versions, or pages.

You can use wildcards in all fields of the directory and drawing names. An asterisk matches any string. A question mark matches any single character.

Some command examples are
:

DIR	Lists all drawing names in the current directory
DIR <*>	Lists all active directories (but no drawing names)
DIR <time>*	Lists all drawing names (parts) in the TIME library
DIR <*>*	Lists all drawing names in all active directories and libraries
DIR ls*	Lists real file name and directory type for current directory.
DIR *.*	Lists all drawings.

Related Commands

IGnore
Library
Use

Display

Syntax

```
DISplay   { Name | Value | Both | Invisible | Default | scale_factor |
Center_justified | Left_justified | Right_justified | Heavy | Thin | Pattern
pattern_number | Filled | Open }{ point . . . | group_name . . . }
```

Description

This command changes the way objects or groups are displayed on a drawing. Any change made with the DISPLAY command remains in effect until another DISPLAY command is used to change it again.

To change the display of a single object, use the left mouse button. To change the display of a group, use the middle mouse button or type in the single-letter group name.

Groups can contain any type of object. Group names, options, and point entries can be included in any order and in any combination, except that the first argument MUST be a command option.

The command options are described below:

- NAME, VALUE, BOTH, and INVISIBLE determine the way properties are displayed on the drawing. Although a property consists of a name and value pair, usually only the value is displayed when a property is added to a drawing. These options allow you to display the name alone, the value alone, both, or neither.
- DEFAULT and SCALE_FACTOR determine the size of text displayed on the drawing.
  - DEFAULT displays text on the drawing using the default text size specified in the *Text* tab of the *Concept Options* dialog box. For example, if the default text size is `0.082` inches, the size of the selected text on the drawing will be set to 0.082 inches.
  - You specify a scale_factor to enlarge or reduce the size of the text on the drawing. For example, if the default text size specified in the *Text* tab of the *Concept Options* dialog box is `0.082` inches, the `DISPLAY 2` command will enlarge the size of the selected text to `0.164` inches.

Note: To display the size of a string, type SHOW SIZE and point to the string. Use SET SIZE to alter the default size of added text.

- A text string added to a drawing is defined by a vertex at the lower left corner of the string. To change the justification, use DISPLAY CENTER_JUSTIFIED, DISPLAY RIGHT_JUSTIFIED, or DISPLAY LEFT_JUSTIFIED.
- HEAVY, THIN, and PATTERN change the way an existing wire appears on a drawing. Heavy makes the wire thicker making it look like a bus. Thin returns a heavy wire to the default wire thickness. Pattern changes a wire to one of six patterned lines. Pattern 1 is a filled line (the default); patterns 2-6 are a variety of dotted and dashed lines. In a LOGIC drawing, the entire net changes. In a SYMBOL or DOC drawing, only the wire segment specified by the cursor changes.
- FILLED and OPEN change the display of dots already added to a design.
- Open dots scale when the ZOOM or SCALE command is used; filled dots do not. The SET DOTS_FILLED command adds dots to the drawing filled by default.

Dot

Syntax

```
Dot point...
```

Description

This command adds dots to drawings to indicate connection points. Dots are used in logic drawings to indicate that lines crossing one another are connected. By default, lines crossing are not connected unless dotted. Wires joining at a 'tee' are connected, even without a dot. Dots are used in symbol drawings to indicate pin connection points.

Dots can be filled or open. By default, all added dots are open. To change to filled dots, type SET DOTS_FILLED. To fill an open dot, type DISPLAY FILLED and point to the dot.

AUTO DOTS places a dot at all connection points in a logic drawing. AUTO UNDOT automatically removes all dots except those at the intersections of four wires.

AUTO UNDOT removes all existing dots in the drawing.

Echo

Syntax

```
Echo message
```

Description

this command displays messages from a script file on the screen. This allows you to track the progress of a script and is useful for debugging purposes.

Enter the message on the same line as the ECHO command.

Edit

Syntax

```
Edit {[<directory>][drawing][.[type][.[version][.[page]]]]|<cr> | point}
```

Description

This command displays an existing drawing to be edited or creates a new drawing.

EDIT <cr> accesses the *View Open* form that allows you to open an existing drawing for editing. The *View Open* form can be turned off by typing SET EDITFORM OFF.

To edit a drawing directly from the command line, type EDIT and the drawing name. <DIRECTORY> is the directory where the editor is to search for the drawing. If not specified, each directory in the list is searched until a drawing by that name is found. The directory name must be enclosed in angle brackets. DRAWING is the name of the drawing to edit. If the specified drawing is found, it is displayed on the screen. If it is not found, the system creates a new drawing by that name in the current library when you write the drawing.

The default value for both version and page is 1. Page specifications for symbol drawings are ignored, but each symbol can have multiple versions. Other drawing types can also have multiple versions and pages.

You can edit a second drawing without writing the current drawing. EDIT saves the first drawing, along with any changes, in a temporary file before bringing in the new drawing. If you edit the first drawing again, EDIT displays the modified version from temporary storage. The SHOW HISTORY command lists all drawings that have been edited during the current session and states whether they have been modified.

The EDIT command also allows you to examine the drawings associated with symbols on the screen. By default, the SYMBOL.CSS file of a hierarchical symbol is edited when you select the symbol from the current drawing. For example, to edit the logic associated with a SUBTRACTOR symbol in the current drawing, type EDIT and point to the symbol with the left button. The current drawing is placed in temporary storage, and the drawing SUBTRACTOR.SYM.1.1 is displayed for editing.

Related Commands

Get
Return

Error

Syntax

```
Error <cr>
```

Description

This command locates and displays each error detected by the CHECK command. It draws a blinking highlighted rectangle at the location of the error and displays a message describing the error.

Exclude

Syntax

```
EXClude [group_name|Mpoint|DEFault][option...][selection ...| group_name]
```

where option is:

```
BOdies | WIres | PRoperties | NEts | Connections
```

and selection is:

```
Lpoint | Ctrl+Rpoint | Mpoint
```

Description

This command removes items or groups from the current group.

- If the first argument is a single-letter group name, the group will become the current group. Alternatively, click the middle mouse button on a highlighted group to make it the current group. If a group is not specified, or the word default is provided, the most recently created group will remain the current group.
- To remove individual objects, click the left mouse button or press *Ctrl* and click the right mouse button.
- To remove previously-defined groups, click the middle mouse button on a highlighted group or enter the single-letter group name.
- Option flags allow the user to remove types of objects in a group. Options are applied to the objects in the initial current group.

  Specifying BODIES, WIRES, or PROPERTIES removes all occurrences of the specified type from the current group. NETS is the same as WIRES. Specifying CONNECTIONS removes all symbol pins (but not the symbol origins) from the current group.

Related Commands

Find
Group
Include
Select

Exit

Syntax

```
EXIt <cr>
```

Description

This commands terminates an editing session. The editor displays a message if there are unwritten changes to any drawings in the current editing session and asks you if you really want to quit. You must answer Y or YES to exit. Any other response aborts the command.

The QUIT command is the same as the EXIT command.

Related Commands

Quit

Filenote

Syntax

```
FILenote {file_name point| <cr>}
```

Description

This command includes a named text file in a drawing at the specified point. POINT is the position in the drawing to add the text. When the file is added, each line in the file is converted into a note that can be individually moved, copied, deleted, or changed. Empty lines in the file are ignored. To include a blank line in the note, type a space on the line in the file.

FILENOTE <cr> brings up a file browser. Select the file and click the position on the drawing where you want to place the note.

Find

Syntax

```
FINd pattern
```

Description

This command searches the current drawing and places all objects that match a specified pattern into a group. A pattern can match symbol names, notes, property names, property values, or signal names. You can search for properties by specifying both name and value separated by an equal sign.

Wildcards are allowed in a pattern. An asterisk matches any number of characters, and a question mark matches any single character. FIND is not case-sensitive.

All items found with the command are placed in a list. You can step through the list items using the NEXT command. This command places a blinking highlighted rectangle around each item on the display so that it can be changed or deleted.

By using the SET NEXTgroup command before the FINd command, you can add the results of the FINd operation to the specified group. Commands to do this are:

```
SET NEXTgroup <groupname>
FINd pattern
```

Examples

- To find all `ls04` components on a drawing and add them in group A, run the following console commands:

```
set nextgroup A
find ls04
```

- To find all objects on a drawing that start with the letter `mem` and add them in group A, run the following console commands:

```
set nextgroup A
find mem*
```

Related Commands

Group
Exclude
Include
Select
Set

Get

Syntax

```
Get { [<directory>][drawing][.[type][.[version][.[page]]]] }
```

Description

This command replaces the current copy of a drawing with the version stored on the disk. The fresh copy of the drawing replaces any previously read (and perhaps modified) version in the editor. GET is useful while editing a drawing if you want to discard the current work and go back to an earlier version.

Related Commands

Edit

Gotosheet

Syntax

```
GOtosheet N
```

where `N` is the page number in a hierarchical design.

Description

This command allows you to go to a specific page in a hierarchical design.

This allows you to easily refer to a page in Concept HDL against a plotted page or a cross-reference report.

- When you plot a schematic page, the page number of the schematic page is plotted if you have added the custom text variables for page numbers on the schematic page. For more information, see Displaying Page Numbers in a Schematic on page 359.
- When you perform cross-referencing on a design, the cross-reference reports display the schematic page numbers.

Enter the page number displayed on the plotted page or the cross-reference report to go to the page in Concept HDL.

Grid

Syntax

```
GRId {<cr>|[ON][OFf][Dots][Lines] [grid_size grid_multiple]}
```

Description

This command alters the grid display. A grid helps you place objects and ensure wire alignment and pin connections. The GRID command options can be used to turn the grid on or off, change the display to solid lines (the default) or dotted lines, and alter the default spacing. GRID <cr> toggles the grid on and off or you can specify ON or OFF. The current values of the grid spacing are displayed on the status line at the bottom of the screen.

Grid_size specifies the separation of the grid lines. Grid_multiple indicates how many lines of the grid are skipped before the next line is displayed. The default value for LOGIC drawings is 5 (2 for SYMBOL drawings). You can specify a positive integer to change the default grid multiple. Specify 1 to display every line, 2 to display every other line, and so on. Be aware that

if you change the grid size, objects that were previously on a grid location may now be off-grid and wires may not be connected even if they appear so. (This is why you should use the right mouse button to connect wires to pins and other vertices.)

See also the SET command to change the default editor values.

Related Commands

Set

Group

Syntax

```
GROup [group_name|DEFault][type...] {selection...| group_name | ALL}
```

where type is:

```
BOdies | PRoperties | NOtes | WIres | DOts.
```

and selection is:

```
{Lpoint Lpoint...Ctrl+Rpoint} | Ctrl+Rpoint | Mpoint
```

Description

This command allows the user to draw a polygon to specify the boundaries of a group. The group is defined as a collection of objects.

- If the first argument is a single-letter group name, that group will become the current group. If a group name is not specified, or DEFault is specified, a single-letter group name will be automatically assigned to the group that is created.
- The group selection can be restricted to a specified type or set of types by providing one or more of the type arguments.

  For example, the command `group A bodies properties` includes only the components and properties among the objects you have selected for grouping in group A, even though there are notes, wires or dots among the objects you have selected for grouping.
- You can use the mouse to draw a polygon around the objects to be grouped. Click the left mouse button to start the line or to change the direction of the line. Complete the polygon by pressing *Ctrl*+right mouse button when the cursor is near the starting point. You can draw additional polygons to include other objects in the group.

- Press *Ctrl* and click the right mouse button to include individual objects in the group.
- Click the middle mouse button on another highlighted group to include its contents in the current group.
- Use the name of a previously created group to include its contents in the current group.

  For example, the command `group C A` includes the contents of group A in group C.
- Use ALL to select all objects in the current drawing.

  For example, the command `group A bodies all` includes all the components in the current drawing in group A.

The console window displays the group name and the number of bodies, properties, notes, dots, and wires in the group.

Related Commands

Auto
Find
Include
Exclude
Select

Hardcopy

Syntax

```
hardcopy [scale] [[<library>]cell.view.version.page]
```

Or

```
ha [scale] [[<library>]cell.view.version.page]
```

For example:

hardcopy 2 notebook.sch.1.* scales the drawing twice and prints all the pages of version 1 of the schematic named notebook.

ha A not*.sch.* scales the drawing to a size A page and then prints all the pages of all the versions of all schematics that start with not.

Description

This command sends drawings to a plotter to produce a printed output. Many types of plotters are supported. Use the SET command (SET <plotter_name> or SET PLOTTER <plotter_name>) to specify the type of plotter.

The user can scale a plot to fit the required size by specifying a scale_option. There are two types of scale_options:

A, B, C, D, E specify various page sizes

scale_factor is a number to scale the drawing from the normal size

HA <cr> plots the current schematic at a scale of 1

To print several drawings with a single HARDCOPY command, use an asterisk to match any number of characters.

For example:

HA A -- plots the current drawing on a size A page

HA C *.logic* -- plots all LOGIC drawings in the current directory on size C pages

HA 1 mux box -- plots all drawing types for the drawing mux box (LOGIC, SIM, TIME, SYMBOL, etc.)

HA 1 <100k>100112.symbol* plots the 100112 part from the 100k library with the default drawing size

See also the SET command to change the plotter type, line width, and font, and to plot timing as immediate or deferred.

Related Commands

Set

Highlight

Syntax

```
HIGhlight [Net | PArt | PIn | Any] pt
HIGhlight [Net | PArt | PIn | Any] object_name
```

Net, Part and Pin are used to specify the object type you want to highlight. You may use Any if you want all selected objects to be highlighted. This is also the default argument if you do not specify any object type
.

pt	To pick a component to be highlighted, point to the object and press the left mouse button. Select a group by pointing to the desired group and pressing the middle mouse button.
object_name	If you prefer typing in the name of the object, you may do so by using the second version of this command.

For nets, the object name is the signal name of the net, for example, FOO.

For parts, the object name is the value of the PATH property attached to the component, for example, 7P.

For pins, the object name is the value of the PATH property attached to the component to which the pin belongs, followed by a period ("."), followed by the name of the pin, for example, 7P.A<SIZE-1..0>.

Wild cards such as "*" and "?" may be used in specifying the object name, for example, FOO*, 7*P, 7P.A*.

Description

The Highlight command is used to select objects in one tool and for operations by other tools in the system. For example, you could type in the `open` command in the simulator and then use the `highlight net` command in Concept HDL, to open a particular signal. Highlight is also used to allow the user to co-relate the same nets, parts and pins in the system. Thus, you could have your PCB layout tool and Concept HDL up at the same time and use the highlight part command in Concept HDL to highlight the component in both Concept HDL and the PCB layout tool.

If an object is already highlighted and you pick the object in the highlight pt version of the command, the object gets unhighlighted in the system. Thus, if you have selected a whole set of nets to open in the simulator, you could just double-click on the nets you want to select in Concept HDL and the net will be selected, but will not be left highlighted. However, if you type in the name of the object, it will be highlighted, even if it was already highlighted.

Related Commands

Dehighlight
Unhighlight

IGnore

Syntax

```
IGnore {directory_name | library_name | * | <cr> }
```

Description

This command causes a specified directory or library and all its drawings and symbols to be deleted from the active search list. IGNORE <cr> ignores the current directory. IGNORE * ignores all the directories in the current search list.

To list all of the current directories, type DIRECTORY <*>.

Wildcards are allowed in these names. If more than one directory matches the pattern, each one is ignored.

IGNORE prompts you to confirm that a directory is to be removed from the search list. Click *Yes* ignore the directory. If the current drawing contains a part from the ignored directory, that part is deleted from the screen and either turned into a placeholder or replaced if there is a part by the same name in another active directory.

Include

Syntax

```
INClude [group_name|Mpoint|DEFault][option...][selection ...| group_name]
```

where option is:

```
BOdies | WIres | PRoperties | NEts | Connections
```

and selection is:

```
Ctrl+Rpoint | Mpoint
```

Description

This command adds items or groups in the current group.

- If the first argument is a single-letter group name, the group will become the current group. Alternatively, click the middle mouse button on a highlighted group to make it the current group. If a group is not specified, or the word default is provided, the most recently created group will remain the current group.

- To add individual objects, click the left mouse button or press *Ctrl* and click the right mouse button.
- To add previously-defined groups, click the middle mouse button on a highlighted group or enter the single-letter group name.
- Option flags allow the user to include types of objects in a group. Options are applied to the objects in the initial current group and to any additions made to the current group by the include command.
  - BODIES or WIRES include all bodies or wires that have properties already in the current group.
  - PROPERTIES include all properties attached to objects already in the related group.
  - NETS include all nets of wires attached to bodies or wires already in the related group.
  - CONNECTIONS include all the objects connected to the pins of any symbol already in the chosen group.

Related Commands

Exclude
Find
Group
Select

Library

Syntax

```
LIbrary {library_name | <cr> }
```

Description

This command adds the specified library to the search list.

LIBRARY <cr> accesses the *Search Stack* form that allows you to add or delete libraries in the active search list. The *Search Stack* form can be turned off by entering the command SET LIBFORM OFF.

To add a library directly, enter the LIBRARY command and the library_name value directly on the command line. The last library added to the list is searched after any previously specified directories are examined.

Related Commands

Directory
IGnore
Use

Loadstrokes

Syntax

```
LOADStrokes {strokes_file | <cr>}
```

Description

This command loads a user-defined strokes_file. Strokes are user-defined line drawings you create with the mouse. You can customize command entry in the editor by using strokes.

When you access the editor, a set of default strokes is initially read from the file <installation_directory>/tools/fet/concept/concept.strokes. You can specify a file containing your own strokes by issuing the LOADSTROKES command at the keyboard or including the command in your startup.concept file.

To draw a stroke, press and hold the right mouse button and drag to form the required stroke. If the stroke you enter does not match an existing stroke, an error message is produced. If you produce a valid stroke, the related command is executed.

Strokes must be entered in the same direction as they were created. This allows you to have two strokes that look the same but that are bound to different commands. For example, you may have two different strokes that both appear as diagonal lines but are bound to different commands. The difference is that one stroke is drawn from upper left to lower right, and the other is drawn from lower left to upper right. When you use the editor to view strokes, a cross signifies the starting point of the stroke.

To create your own strokes_file, or edit an existing strokes_file, you use the Stroke Editor. This system utility is used to create strokes for many applications.

LOADSTROKES <cr> brings up the File Browser form. You can then select the strokes_file from the form.

Related Commands

Strokefile

Mirror

Syntax

```
MIrror point
```

Description

This command creates a mirrored version of a selected symbol. If editing a symbol drawing, this command will not mirror all lines and arcs in the drawing about the Y axis. Justified text is shifted from left to right or right to left in the mirrored version. No other rotation is done.

This command should be used with caution, especially with bodies with unmarked pins, such as merge bodies. Reversing the bits causes subtle, hard-to-find errors in the design.

See SET LEFT/RIGHT and DISPLAY LEFT/RIGHT for justifying text, and the ROTATE, SPIN, and VERSION commands.

Related Commands

Display
Rotate
Set
Spin
Version

Modify

Syntax

```
MODify point
```

Description

This command modifies the selected part. Concept HDL displays the Physical Part Filter dialog box which displays a list of all parts satisfying the selection criteria based on the properties attached to the selected component.

If the user selects a primitive from the popup, the package type corresponding to the primitive is used to seed the PACK_TYPE property in the PPFF.

The old part is modified, the properties on the old part is now a union of the properties that existed on the old part and the properties on the new part. If the old part and the new part have the same property name but different property values,

- if multiple properties with this name can be legally attached, a new property identical to the one on the new part is attached to the old part

- otherwise, the value of the property on the old part is modified with the value of the property on the new part.

All error messages are displayed in the Concept HDL console window.

Move

Syntax

```
MOVe {source_point destination_point|group_name destination_point}
```

Description

This command moves objects from one position to another in the current drawing or between drawings in different viewports. MOVE operates on groups or individual objects. Source_point is the object to move. Group_name is the name of a group to move. Destination_point is the new position of the group or object. Properties (excluding PATH properties) attached to objects are moved with objects. Properties can also be moved independent of objects. To undo the effect of moving objects from one viewport to another viewport, an undo command must be entered in each viewport.

To move a single object, type MOVE and position the cursor on the object to move. Press the left button to pick up the object that is nearest to the cursor (regardless of the grid setting) or the right button to pick up an object's vertex nearest the cursor. A vertex is defined as a symbol origin, symbol pin, a wire end, or a note origin. The right button is useful for moving bodies or off-grid objects. Move the object to its new location and press the left button to place the object on the grid point nearest the cursor or the right button to attach the object to the nearest vertex. Note that when using the right button for the source_point, any visible object is selected. The right button as a destination_point only considers symbol pins, symbol origins or wire ends as attachment points.

To move an object from off-grid to on-grid, select the object with the right mouse button and place it with the left mouse button for the nearest grid point or with the right mouse button for

the nearest vertex. If an off-grid object is selected with the left mouse button and placed with the left mouse button, it will remain off-grid.

To move a defined group, specify the name of the group or press the center button to select the group to move.

MOVE preserves electrical connectivity when there are electrical connections (wires) leading to moved objects or groups. You can automatically re-route a wired part you have moved to another area of your design. You can move the part into place with the wires connected directly or orthogonally, or without wires as follows:

The first click of the middle button (while dragging) changes the shape of the wire from orthogonal to direct.

The second click of the middle button detaches the part from any wires and allows you to move the part freely.

The third click of the middle button re-attaches the wires and lets you drag the object the usual way until you place the object with the left button.

The wires are automatically re-routed only when you place the part with the wires connected directly (non-orthogonally) and the SET option AUTOROUTE is on (SET AUTOROUTE ON).

To move a whole wire and any attached object around the screen, select the middle of the wire. To lengthen a wire, select the outer third of the wire. When a wire is attached between two objects, you can move the wire and one object independently of the other object by selecting the wire nearest the object you want to move.

Next

Syntax

```
NExt  <cr>
```

Description

This command displays the items located by the FIND command. The NEXT command traverses the list of items found by the FIND command and draws a blinking highlighted rectangle around the item. You can perform an operation on the object and then issue the NEXT command to proceed to the next item. You can step through the list only once.

NEXT cannot be used after the CHECK command. Use ERROR after the CHECK command.

Related Commands

Check
Error
Find

Note

Syntax

```
Note text_line... point...
```

Description

This command adds text strings to a drawing. Notes are text strings that appear on the drawing but do not affect the evaluation of the drawing. They are used to document a drawing. There are two ways to add notes to a drawing:

- Specify the points on the drawing where the notes are to be located and then type in the text. Press <cr> after each note to position each note on the drawing. As long as there are points remaining, the editor interprets entered text as notes to the drawing.
- Type in each line of text first and press <cr>. (You can enter several strings before placing them.) Then use the cursor and the left button to indicate where each note is to appear on the drawing.

Place quotes around notes beginning with an opening parenthesis. Notes within quotes are not interpreted as commands.

Related Commands

Filenote

Paint

Syntax

```
PAInt {color_name{point|group_name}|DEFault{point|group_name}} <cr>
```

Description

This command assigns selected colors to specified groups or objects. PAINT <cr> accesses the Paint form. You can use this form to select colors. You can also enter the color_name on the PAINT command line, point to an object and press the left button. Select a group by entering the group_name or by pointing to the required group and pressing the center button.

PAINT DEFAULT paints objects or specified groups in their preset default colors. Use the SET COLOR commands (interactively or in the startup file) to establish default colors for the objects in your drawings. On a monochrome display, use the SHOW COLOR command to see what color an object is currently painted.

There are 16 available colors

- AQUA
- GREEN
- PINK
- SKYBLUE
- BLUE
- MONO
- PURPLE
- VIOLET
- BROWN
- ORANGE
- RED
- WHITE
- GRAY
- PEACH
- SALMON
- YELLOW

Pause

Syntax

```
PAUse
```

Description

this command temporarily interrupts the editor until you press a key. PAUSE is useful in demos and scripts.

Pinnames

Syntax

```
PINNames point
```

Description

This command adds a PIN NAMES symbol to a schematic drawing that defines the functional circuitry of a symbol drawing. This is used in hierarchical design and in library development.

The PIN NAMES symbol is added to an unused area of the schematic drawing. Concept HDL automatically attaches the names of the pins on the corresponding symbol drawing to the PIN NAMES symbol in the schematic drawing, and appends a `\I` suffix (scope = interface) to each signal name. The signal names can then be reattached to signals in the schematic drawing. The use of the PIN NAMES symbol eliminates the need to retype the signal names and reduces the chances of mislabeling signal names or omitting the interface scope (`\I`) signal property.

Pinswap

Syntax

```
PINSwap {point1 point2 | pin_number point }
```

Description

This command swaps the pin number defined to be in the same pin group. This command can only be used after initial pin number assignment using the SECTION command. Also, pin

swapping can only occur between pins that have been defined in the library as swappable. For example, it may be legal to swap the two input pins of a NAND gate, but not the input and output pins of the gate.

There are two ways to swap pins:

- Type PINSWAP and point to the two pins to be swapped.
- Type PINSWAP, type in a new pin number, and then point to an existing pin. The selected pin is swapped with the pin having the pin number you specified.

The properties attached by the PINSWAP command cannot be changed, they can only be deleted and moved. Once pins on a part have been swapped, the part cannot be resectioned using the SECTION command.

The PINSWAP command also swaps sections within HAS_FIXED_SIZE parts.

Related Commands

Backannotate
Section

Plot

Syntax

```
Plot
```

Plots the currently opened drawing

```
Plot cache
```

Plots all pages of the schematic named cache.

```
Plot cache.sym.1.1
```

Plots the symbol view of cache

```
Plot cache.sym.1.2
```

Plots page 2 schematic of cache

```
Plot cache.sch.1.*
```

Plots all pages of version 1

PPTAdd

Syntax

```
PPTAdd {"path"[",path"];}
```

Description

When using the component selection in the physical mode, PPTADD provides the search path to locate the ppt files to be used.

The list of paths are pushed on to a stack. The last one specified ise the first one searched.

This command replaces the SET PPTPATH option.

Related Commands

Add
PPTDelete

PPTDelete

Syntax

```
PPTDelete {"path"[",path"];}
```

Description

To remove a path from the list or stack of path strings used in physical component selection. It does not affect the stack order.

Related Commands

Add
PPTAdd

PPTEcho

Syntax

```
PPTEcho
```

Description

When using the component selection in physical mode, PPTECHO lists the search path which will be used to locate the ppt files.

Related Commands

Add
PPTAdd
PPTDelete

Property

Syntax

```
Property {attach_point location_point name_and_value |  name_and_value
attach_point  location_point  | attach_point  name_and_value  location_point}
```

Description

This command attaches a property name and a value to a specified vertex of an object. Properties allow you to associate information with selected objects on a drawing. The information is passed to other design programs for processing and analysis. A property consists of a name-value pair that is attached to an object (a symbol, pin, wire, or signal name). Operations on groups are not performed using the property command. Instead, use AUTO PROPERTY.

A property name can be any string of alphanumeric characters and underscores, provided that the first character is an alphabetic character. A property name cannot contain any spaces or punctuation marks except the underscore.

A property value can be any string of text up to 255 characters, including spaces and punctuation marks.

There are two ways to assign properties.

- Type PROPERTY <cr>.

  Select the objects where properties are to be attached. Use the left button for objects. Select as many objects as the number of properties you want to attach. Type the name and value of the property, separated by a space or an equal sign. Press <cr> after each property entry. The properties are attached to the selected objects in the same order as the initial selection.

- Type PROPERTY <cr>.

  Type the name and value of the property, separated by a space or an equal sign. Press <cr> after each property entry. Then specify the location on the drawing where the text of the property value should appear. Select as many objects as the number of properties you entered. The properties are attached to the objects you select in the same order as the initial property entries.

Each property attached to a given object, except the SIG_NAME property must have a unique name. If a newly entered property has the same name as a property currently attached to that object, the new property value replaces the old property value.

When a property is added to a drawing, only the property value appears. The SHOW PROPERTIES command temporarily displays the names and values of all properties on a drawing. The DISPLAY command changes the permanent display of property name and value pairs. The SET PROP_DISPLAY command controls the default display of added properties.

Quit

Syntax

```
QUIt  <cr>
```

Description

This command terminates an editing session. The editor displays a message if there are unwritten changes to any drawings in the current editing session and asks you if you really want to quit. You must answer Y or YES to quit. Any other response aborts the command.

The EXIT command is the same as the QUIT command.

Related Commands

Exit

Reattach

Syntax

```
REAttach text_point  attach_point
```

Description

This command reattaches properties (including signal names) from one object to another. For example, you can use the REATTACH command to attach a property from the input pin to the output pin of a device.

To reattach a property, type REATTACH and select the property. A line is drawn from the property to the current cursor position. Specify the new attach_point for the property. Use the MOVE command to position the property at its new attachment point.

Recover

Syntax

```
RECover <recover_log_file>
```

Description

This command is used for recovering drawings that were being edited when Concept HDL or your system crashed.

Everytime you start Concept HDL, a temporary directory is created in the `<project_directory>/temp` directory. By default, `xxnedtmp` is the name of the temporary directory. If the `xxnedtmp` directory already exists, a `xxnedtmp1` directory is created. If these two directories already exist, `xxnedtmp2` is created, and so on. An undo log file for each drawing is stored in this directory. The name of the undo log file for the first drawing edited is `undo1.log`. The second drawing's undo log file is `undo2.log`, and so on.

Note: The temporary directory (for example, ./`xxnedtmp`) and all of its files are deleted if Concept HDL terminates normally.

An example illustrating the use of the `recover` command is:

```
RECOVER temp/xxnedtmp1/undo2.log
```

The recovered drawing is given a unique name (for example `RECOVER1.SCH.1.1`) and is only saved in the memory (not on disk). You should use the `diagram` command to change the drawing name and use `write` command to write the drawing out to the disk.

Related Commands

Diagram
Write

Redo

Syntax

```
REDo  <cr>
```

Description

This command reverses the last UNDO command. The system keeps a list of operations performed during the current editing session in a log file. The UNDO and REDO commands perform their functions according to this log file.

Remove

Syntax

```
REMove [<directory>][name][.[type][.[version][.[page]]]]
```

Description

Deletes a drawing from a design directory. REMOVE allows only one argument at a time. Repeat the procedure to delete additional drawings. If no directory is given, REMOVE searches for the specified drawing in the currently active design directory.

To delete a drawing, type REMOVE and the name of the drawing to be deleted, and press <cr>. The editor displays the names of the files to be deleted and asks you if you really want to remove the files. You must answer Y or YES to remove them. Any other response aborts the command. The directory entries are deleted, and the files are purged.

Wildcards are allowed in drawing_name. A question mark matches any single character, and an asterisk matches any number of characters. If only the drawing_name is specified,

REMOVE deletes all drawing types (SYMBOL, LOGIC, SIM, and so on), versions, pages, and files (ASCII, binary, dependency, and connectivity) of the specified drawing in the directory.

Replace

Syntax

```
REPlace {symbol_name point | symbol_name <cr> group_name}
```

Description

This command substitutes one part for another. The default version of the symbol is 1. Specify the version number to replace another version of a symbol.

There are several ways to use the REPLACE command.

- Type the name of the replacement part. Then use the cursor to point to the symbol or bodies to be replaced.
- Use the FIND command to group all the occurrences of a symbol to be replaced. Then, use the group_name option with the REPLACE command to globally change all the occurrences of the symbol. A message displays the number of bodies that are replaced.

Pin properties are reattached if a pin name on the new part is the same as a pin name on the first part. If the pin names do not match, the pin property becomes a symbol property.

All properties are retained except those generated by the BACKANNOTATE, SECTION, and PINSWAP commands. Unnamed signal names attached to the symbol are deleted. All default properties that have a value of ? receive the value of the property with the same name on the replaced symbol (if one exists). Wire connections to the original part are retained only if the pins are in the same location. The rotation of the original symbol is preserved when the symbol is replaced.

Return

Syntax

```
RETurn <cr>
```

Description

This command returns to the previously edited drawing. If the current drawing is modified but not written, the system saves a copy of the drawing before returning to the previous drawing.

The SHOW HISTORY command lists the drawings that you edited during the current session.

The SHOW RETURN command lists the drawings that the RETURN command will return to in the order that they will be accessed.

Rotate

Syntax

```
ROTate point
```

Description

This command rotates a symbol or text string by 90 degrees, with mirrors at 180 and 270 degrees. When a symbol is rotated, all notes and properties are also rotated and translated. You can then act on the properties independently.

To rotate a symbol or text string, type ROTATE and then point to the object to rotate. Each time you press the button, the part rotates 90 degrees. In the 90 -degree rotation, symbol notes are rotated 90 degrees to the left in their original justification.

Rotating some parts 180 degrees reverses the order of the pins. This can cause subtle errors in your designs if pins become incorrectly wired. To avoid this, a 180-degree rotation of a part becomes a mirror of a 0-degree rotation (about the Y axis). A 270-degree rotation of a part is a mirror of a 90-degree rotation (about the X axis). To get the other two rotations and the other two mirrors, use the MIRROR command to create another version of the device.

See SET and DISPLAY for justifying text.

Related Commands

Add
Display
Mirror
Set
Spin

Route

Syntax

```
ROUTE point point
```

Description

This command draws a wire connecting two selected points. The ROUTE command connects two points by drawing a series of orthogonal line segments between them. If it cannot determine a route, it draws a diagonal line directly between the two points. ROUTE will not run a wire through any existing objects or vertices.

To select the nearest pin or wire vertex for a ROUTE point, use the right button to select the point. Use any other button to select the nearest grid point.

Related Commands

Broute
Bwire
Wire

s2l

Syntax

```
s2l design
```

Description

This command runs the s2l (short2long) command on the current design. To update only the current page, type

```
s2l
```

The s2l command should be run on designs that have been upreved from SCALD to Concept HDL having property names that exceed 16 characters.

Before you run this command, you should execute the following two commands:

```
uprev design
uprev_write
```

In SCALD designs, property names in Concept had a limit of 16 characters. For a property name that has more than 16 characters, Concept HDL assigns a new property name with a shortened version and the original property name (over 16 characters) as its value. The original value, which has more than 16 characters, is entered by the user in <*your_install_dir*>/tools/fet/pxl/allegroprp.dat.

When Packager-XL is run on the SCALD design, it replaces the shortened property name with the original property name after finding it in allegroprp.dat.

For example, the property ELECTRICAL_CONSTRAINT_SET is shortened to ELECTRICAL_CONST by Concept HDL. Concept HDL also assigns ELECTRICAL_CONSTRAINT_SET as the value for the ELECTRICAL_CONST property. When Packager-XL is run on the design, it converts ELECTRICAL_CONST to ELECTRICAL_CONSTRAINT_SET and passes the design to Allegro.

In HDL, Packager-XL does not perform this conversion. Instead, this functionality of Packager-XL is handled by the s2l command entered in the Concept HDL console command window.

The character limit for a property name in Concept HDL is now 31.

Scale

Syntax

```
SCAle {point1 point2 drawing_name | drawing_name point1 point2}
```

Description

This command smashes a drawing and includes it in the current drawing. Point1 and point2 indicate the size of the rectangle where the smashed drawing will be placed. Drawing_name is the name of the drawing to smash. All bodies are turned into wires, arcs, and text. SCALE is useful for creating documentation drawings and new bodies.

When a drawing is smashed, all connectivity information is lost. The drawing can no longer be interpreted by the Compiler.

Related Commands

Smash

Script

Syntax

```
SCRipt { file_name | <cr> }
```

Description

This command performs the commands listed in the specified text file. Script files let you change the default editor behavior. SCRIPT allows you to operate in the batch mode using the same syntax as if you typed in the command. You can use the mouse to enter points, or you can specify the X-Y coordinates in the script file.

The syntax for specifying the points is:

```
(X Y)
```

where X and Y can range from +16000 to -16000.

For example, the command `vpdelete (0,0)` will delete the viewport that has the coordinates `(0,0)`.

You can configure a script to accept input during execution by including user input tokens in a script. User input tokens must be placed at the beginning of a new line. When the editor sees a user input token in a script, it highlights a menu button with the name of the editor command being executed. There are two user input tokens:

- $< When the editor encounters this token in a script, it prints from the token to the end of the text line as a prompt in the message window, and then waits for one item of input. The input can be a typed line, a function key press, a mouse action, or a *Ctrl+C*. You cannot use a <cr> as a response to a user input request.
- $; This token also prints from the token to the end of the text line as a prompt and awaits input. This token accepts and interprets inputs until you enter a semicolon. If this token is included, the editor follows the prompt with the message "Type ; when done with user input."

To abort a script, press Ctrl+C. To abort at a user input token prompt, type a semicolon.

SCRIPT <cr> brings up the File Browser. The user can then select the names of the script file from the form.

The SCRIPT command will not recognize GED commands that begin with the word "FORCE" (such as FORCEPROP, FORCEADD, FORCEBUB, FORCENOTE, and FORCEQUIT).

Related Commands

IGnore
Library
Set
Use

Searchstack

Syntax

```
SEArchstack <cr>
```

Description

Accesses the SEARCHSTACK BROWSER form. This form lists the libraries and directories that are currently accessed through the USE or LIBRARY commands.

BROWSE opens or updates the DIRECTORY BROWSER form that lists the contents of a library or directory.

USE places the active directory at the top of the search list and makes it your current working directory. There is no limit to the number of directories and libraries that can be in use at one time.

IGNORE deletes the specified directory or library from the active search list. When you select a directory or library to ignore, the editor prompts you to be sure you want to ignore it. Move the cursor to the message window, type Y or N, and press <cr>. The specified directory or library is removed from the search stack.

LIBRARY accesses the AVAILABLE LIBRARIES form that lists all the available libraries. From AVAILABLE LIBRARIES, you can select any number of libraries to add to the active search list. As you select libraries, the library names appear in the SEARCHSTACK BROWSER form.

Section

Syntax

```
SECtion [pin_number]point
```

Description

This command displays different pin numbers for different sections of a symbol. The SECTION command lets you assign physical part sections to selected logical parts. As you step through the different sections of a symbol, the pin numbers of each section are displayed on the drawing. Sectioning a part automatically assigns path properties to the drawing.

If the logical part selected can be assigned to a section, the pin numbers for the section are displayed on the drawing. If the same part is selected again, the next section is selected and the new pin numbers are displayed. This makes it possible to step through all the different possible sections by pointing to the same part.

To assign a specific section directly, enter a pin_number that uniquely defines the section and then point to the part. This avoids having to step through each section individually.

To remove section information from a part, use the REPLACE command to replace the sectioned symbol with a new copy of the part.

You can section only parts with SIZE = 1 or HAS_FIXED_SIZE characteristics. To assign sections to a HAS_FIXED_SIZE part, point to the pin of the section to be assigned. To swap sections within a HAS_FIXED_SIZE part, use the PINSWAP command.

Related Commands

Backannotate
Pinswap
Set

Select

Syntax

```
SELect [group_name | DEFault][type...]{selection...| group_name |ALL}
```

where type is:

```
BOdies | PRoperties | NOtes | WIres | DOts
```

and selection is:

```
{Lpoint Lpoint} | Ctrl+Rpoint | Mpoint
```

Description

Provides a stretchable rectangle to specify the boundaries of a group. The group is defined as a collection of objects.

- If the first argument is a single-letter group name, that group will become the current group. If a group name is not specified, or DEFault is specified, a single-letter group name will be automatically assigned to the group that is created.
- The selection can be restricted to a specified type or set of types by providing one or more type arguments.

  For example, the command `select A bodies properties` includes only the components and properties among the objects you have selected for grouping in group A, even though there are notes, wires or dots among the objects you have selected for grouping.
- You can click the left mouse button, drag the mouse to define the opposite corners of a stretchable rectangle which contains the objects to include in the group, and click again. You can draw additional rectangles to include other objects in the group.
- Press *Ctrl*+right mouse button to include individual objects in the group.
- Click the middle mouse button on another highlighted group to include its contents in the current group.
- Use the name of a previously created group to include its contents in the current group.

  For example, the command `select C A` includes the contents of group A in group C.
- Use ALL to select all objects in the current drawing.

  For example, the command `select A bodies all` includes all the components in the current drawing in group A.

The console window displays the group name and the number of bodies, properties, notes, dots, and wires in the group.

Related Commands

Auto
Find
Group
Include
Exclude

Set

Syntax

```
SET {option | <cr>}
```

Description

This command establishes the default options for the editor. SET commands can be issued during an editing session or placed in the startup.concept file.

SET <cr> accesses the *Concept Options* dialog box. You can then use this dialog box to set different options. You can close the Concept HDL Options dialog box by typing SET SETFORM OFF in the Console Command window.

The SET arguments shown here are grouped by function. Related options (usually opposites) are listed together and are separated by a slash. The default is shown first.

	Set Command Options
CHANGE	INLINE_TEXTEDIT <ON/OFF> USER_Editor <string>
CHECK	CHECK_Arcs_at_same_loc <ON/OFF> CHECK_Body_place_holders <ON/OFF> CHECK_Hidden_wires <ON/OFF> CHECK_Missing_pins <ON/OFF> CHECK_Net_names_hdl_ok <ON/OFF> CHECK_On_write <ON/OFF> CHECK_PACk_sec_type_props<ON/OFF> CHECK_PARts_at_same_loc <ON/OFF> CHECK_PIN_near_wire_endpt <ON/OFF> CHECK_PINS_at_origin <ON/OFF> CHECK_POrt_names_hdl_ok <ON/OFF> CHECK_PRop_place_holders <ON/OFF> CHECK_SHorted_pin <ON/OFF> CHECK_SIGNAMES <ON/OFF> CHECK_SIGNAME_in_body <ON/OFF> CHECK_SYmbol_names_hdl_ok <ON/OFF> CHECK_TItle_abbrev <ON/OFF> CHECK_TWo_wires_at_pins <ON/OFF> CHECK_Unconn_wires <ON/OFF>
DEPENDENCY MANAGEMENT	ALlegro/NOALlegro

	Set Command Options
DOTS	AUTODot <ON/OFF> DOTS_Filled/DOTS_Open BOdy_dot_radius <integer> LOgic_dot_radius <integer> COLOR_Dot <color>
EDITING DRAWINGS	CLIck_to_type <ON/OFF> PUsh_type OVERview <ON/OFF> BACKground_color <color> STRoke <ON/OFF> USE_PUsh_type <ON/OFF>
FORMS	SETform <ON/OFF> ADDform <ON/OFF> LIBform <ON/OFF> EDITform <ON/OFF> DIRform <ON/OFF> ATTRIBUTES_Dir ATTRIBUTES_Filter
GRID	DECimal/FRactional/Metric DEFAULT_Doc_grid <size> <mult> DEFAULT_Body_grid <size> <mult> DEFAULT_Grid <size> <mult> GRID_OFF/GRID_ON
HDL DIRECT	HDL_Direct <OFF/ON> HDL_Checks <OFF/ON>
MISCELLANEOUS	NEXTgroup Warnonuse BLOck_title_top CAT_path MERgeppt
PAINT SELECTION	COLOR_Arc <color> COLOR_Body <color> COLOR_Dot <color> COLOR_Wire <color> COLOR_Note <color> COLOR_Property <color>

	Set Command Options
PLOTTING DRAWINGS	HPF Options (UNIX only)
	DOUble_width/SINgle_width FOnt hpf_font_name LOCAl_plot/SPOoled_plot papersize <option> PLotter plotter name
	Windows Plotting Options (UNIX and NT)
	wplot_spooled (print to a file) wplot_local (print to a printer) wplot_thin_width (single line width in plots) wplot_thick_width (double line width in plots) wplot_screen (plots contents on screen) wplot_sheet (plots contents on sheet)
	The set wplot_spooled command sets the default plotting method as print to file for the current session. To set this option for a given project in all sessions and specify the name of the generated postscript file, add the following directives manually in the project file (<project_name>.cpm):
	PLOT_TO_FILE ‘NO’
	PLOT_FILE_NAME ‘output.ps’
	Note: The format of the plot file (when you set wplot_spooled) depends on the plotter driver. The formats can be PCL, PS or any other.
PROPERTIES	PIn_size <real> PRop_display <display_type> ROtate <ON/OFF> STICKY_OFF/STICKY_ON AUTOPath <ON/OFF>
SAVE WORKSPACE	SAve_workspace <ON/OFF>
TAP COMMAND	TAP_Add_signal <ON/OFF> TAP_Body <string>
TEXT ENTRIES	CAPSLOCK_OFF/CAPSLOCK_ON SIZe <real> CEnter_justified/LEft_justified/RIght_justified

	Set Command Options
WIRES	MOVE_Orthog/MOVE_Direct ORthog_wire/DIrect_wire AUTORoute <ON/OFF> AUTOHeavy <ON/OFF>
WRITING DRAWINGS	AScii/NOAscii BInary/NOBinary CONN/NOConn CONFirm_write <ON/OFF> DEPendency/NODependency CHECK_On_write <ON/OFF>

set NEXTgroup

This command sets the name of the group you want to create with the Find command.

If a group with the same name exists, this command resets the group and allows you to create a new group with the same name. For example, suppose that group A contains 2 properties. If you run the following console commands on a schematic that has two instances of the `ls04` component, the new group A will contain 2 bodies:

```
set nextgroup A
find ls04
```

set HDL_Direct

This command sets options for the `write` console command. When you save a drawing, the `write` command is executed that writes the drawing onto the disk.

When the HDL_Direct option is on/off, the `write` command writes/does not write the following files in the sch_1 view of the schematic:

- `verilog.v`
- `vhdl.vhd`
- `hdldirect.dat`
- `viewprps.prp`
- `*.sir`

Note: The `hier_write` and `uprev_write` console commands are not affected by the HDL_Direct option. These commands always write all the files onto the disk.

set sticky_on and set sticky_off

When you place a component on the schematic, the properties specified on the symbol for the component become the default properties for the instance of the component. You cannot delete the default properties for the instance.

If you delete a property or modify the property name on the symbol, the property may still be present on the instance of the component as a default property. These are called dangling properties. Concept HDL displays the following error message when it finds dangling properties on the schematic:

```
The default property <property> [with value <value>] is no longer on the body
<symbol_name>. To turn the deleted default properties into non-default properties
type SET STICKY_ON;GET;SET STICKY_OFF.
```

Note: The dangling properties will not be visible on the schematic or in the *Attributes* dialog box.

- If you want the dangling properties to be converted to non-default properties, run the following console commands in the following order:

  ```
  set sticky_on
  get
  set sticky_off
  ```

  The dangling properties become visible on the schematic or in the *Attributes* dialog box. The non-default properties are applicable only to the instance of the component in the schematic. You can delete the non-default properties on the component.

- If you want to delete the dangling properties, do the following:

  a. Run the `set sticky_off` console command.

  b. Run the `write` console command or choose *File > Save* to save the schematic.

Show

Syntax

```
SHow {option |<cr>}
```

Description

This command temporarily displays objects or information on classes of objects. The temporary information is erased when you redraw the screen.

To see all the SHOW options, type SHOW <cr>. The following SHOW options are available:

Attachments	Displays a line between visible properties and the property owner
COLor pt	Displays the color of the selected object The color is displayed in the Concept HDL console window and as ticker text in the status bar.
COOrdinate pt	Gives the editor the location of the specified point in coordinates
DISTance pt pt	Displays the distance between two selected points. ■ Clicking the left mouse button at two grid points displays the distance between the two grid points. ■ Clicking the middle mouse button (in a three button mouse) or *Ctrl*+left mouse button at two points on the screen displays the distance between the two screen points. The distance is displayed in the Concept HDL console window and as ticker text in the status bar.
Group pt	Highlights the group and lists the group name and contents. You can also specify the group by name.
History	Lists all drawings read in during the editor session
Keys	Lists the command string assigned to each function key
Modified	Lists drawings in all viewports that were modified but not saved during the editor session
Net pt	Highlights the selected net and displays the net name. You can also specify the net by name. The net name is displayed in the Concept HDL console window and as ticker text in the status bar.
Origins	Displays an asterisk at each object origin (including text) on display
PIns	Displays an asterisk at each pin on display
PRoperties	Displays the names and values of all properties on the display (including any invisible properties)
PWd	Displays the name of the current project directory The current directory is displayed in the Concept HDL console window and as ticker text in the status bar.
RELease	Lists the date and number of the current version of the editor

RETurn	Lists all the drawings that the editor can return to, in the order of return
Size pt	Displays the size of the selected text string (property name, property value, signal name, note) in inches. The text size is displayed in the Concept HDL console window and as ticker text in the status bar.
VECtors pt	Displays the pin names of the selected symbol

Signame

Syntax

```
SIGname {point signal_name | signal_name point }...
```

Description

This command attaches signal names to wires or pins. There are two ways to attach a signal name.

- Type SIGNAME and point to the location for each signal name. A rectangular box appears at each location. Type the text for the signal name and press <cr>.
- Type SIGNAME and enter one or more signal names. Specify the points to place the signal names on the drawing. The signal name is attached to the wire or pin that is closest to the specified point.

Signal names are handled internally as properties. For example, attaching a signal called BUS ENABLE to a wire is equivalent to attaching a property SIG_NAME=BUS ENABLE, to that wire.

When editing a SYMBOL drawing, signal names are known as PIN_NAME properties. They can be attached only to pin connections.

Related Commands

Busname
Property

Smash

Syntax

```
Smash {point | group_name}...
```

Description

This command breaks a symbol into individual wires, arcs, and notes. Any properties attached to the symbol are deleted. The SMASH command works on individual bodies and on groups.

The SMASH command is useful for creating library symbol drawings. For example, once a 2-input AND gate exists, N-input AND gates can be made by using the following commands:

```
edit N AND.body
add 2 AND <pt>
smash <pt>
```

Attach the N inputs and write the drawing. Because 2 AND is no longer a symbol, the editor writes the drawing instead of producing an error message as it does when a symbol is added to a symbol drawing.

Related Commands

Filenote
Scale

Spin

Syntax

```
SPIn point...
```

Description

This command changes the orientation of text strings and components. Spins are in 90-degree increments (0, 90, 180, and 270). When you spin a symbol, all notes and properties are also spun and translated. You can then act on the properties independently.

The SPIN command does a true rotation of the symbol, as opposed to the ROTATE command, which mirrors for 180 and 270 degrees. Use SPIN with care; allowing 180-degree

rotations of devices may reverse the order of the pins (for example, in mergers). This can cause subtle errors in a design.

Related Commands

Mirror
Rotate
Set
Version

Split

Syntax

```
SPLit point point...
```

Description

This command splits a single wire into two wires or separates two or more objects that are placed at the same location. The currently selected item blinks so you know what you have selected.

To split a single wire into two wires, type SPLIT and select a point along the wire with the mouse. If you want to separate the end of a wire from some objects, point near the end of the wire. If you want to break a wire into two attached segments, point near the middle of the wire. This creates a bend in the wire at the selected location. Select and position the appropriate section of the wire.

To disconnect two or more items at the same location, type SPLIT and select a location. One of the objects at that location is attached to the cursor and can be moved on the screen. Select the original location again to separate the second object. Continue to select objects until the correct item is attached to the cursor. You can cycle through the objects repeatedly. Move the object to its new location and click the mouse to place the object.

Strokefile

Syntax

```
Strokefile {strokes_file | <cr>}
```

Description

This command loads a user-defined custom strokes_file. Strokes are user-defined line drawings you create with the mouse. You can customize command entry in the editor by using strokes.

When you access the editor, a set of default strokes is initially read from the file <installation_directory>/tools/fet/concept/concept.strokes.

You can specify a file containing your own strokes by issuing the STROKEFILE command at the keyboard or by including the command in your startup.concept file.

To use a stroke, press and hold the left or center mouse button and draw the required stroke. If the stroke you enter does not match an existing stroke, an error message is generated. If you draw a valid stroke, the related command is executed.

Strokes must be entered in the same direction as they were created. This allows you to have two strokes that look the same but that are bound to different commands. For example, you may have two different strokes that both appear as diagonal lines but are bound to different commands. The difference is that one stroke is drawn from upper left to lower right, and the other is drawn from lower left to upper right. When you use the editor to view strokes, a cross signifies the beginning of the stroke.

STROKEFILE <cr> brings up the Browser. You can then select the strokes_file from the form.

Related Commands

Loadstrokes

Swap

Syntax

```
Swap point1 point2...
```

Description

This command swaps two properties or two notes. Only two notes or two properties can be swapped, not a note and a property. Default properties and those generated by the PINSWAP, SECTION, and BACKANNOTATE commands cannot be swapped.

System

Syntax

```
System {operating_system_command | <cr> }
```

Description

This command accesses the operating system. To execute a particular system command, enter it on the same line as the SYSTEM command. Without an argument, the editor provides an interactive shell. You are connected to the operating system and can run any operating system commands.

To exit from the operating system and return to the editor, type Ctrl-D or Exit on UNIX-based systems and EXIT on NT-based systems.

Tap

Syntax

```
TAp bus_tap_value point point...
```

Description

This command taps a bit or a set of bits from a bus and wires them to a pin. The bus_tap_value is the bit or set of bits you want to tap off the bus. You can enter any number of bus_tap_value(s) before you start selecting points. There are two modes of this command. In mode 1, you first select the pin you want to tap to and then the bus to tap from. In mode 2, you first choose the bus and then choose the pin.

Mode 1:

1. Enter one or more bus_tap_values -- such as 3..0, 5, 2 etc. separated by CARRIAGE_RETURNs. If you do not enter a bus_tap_value, a question mark is used for the tap bits.
2. Choose the pin you want the tap to go to. Use any mouse button. Now, you have a wire attached to the mouse.
3. If you click at a point with the left mouse button (and the point is not too close to a bus, you will get a kink at the point you clicked and you can keep drawing your wire. If you choose a point very close to a bus, or you use the middle or left mouse buttons, the command will find the closest bus, and draw a wire from your last point to the bus. It also

adds a tap symbol, called "CTAP" by default, between your bus and the wire. The BN property on the tap symbol will be given the appropriate value. If the set option TAP_ADD_SIGNAL is ON, a signal name will be attached to the wire, specifying the bus name the tap has gone to and the bits tapped assuming that the bus you tap from has a vectored name.

4. The next signal name you entered will be used for the next tap (i.e. go back to step 1 or 2).

Mode 2:

1. Enter one or more tap names -- such as 3..0, 5, 2 etc. separated by Carriage Returns If you do not enter a tap name, a question mark will be used for the tap bits.
2. Choose the bus you want the tap from. Use any mouse button.
3. Now, you have a wire attached to the mouse. If you click at a point with the left mouse button (and the point is not too close to a pin, you will get a kink at the point you clicked and you can keep drawing your wire. If you choose a point very close to a pin, or you use the middle or right mouse buttons, the command will find the closest pin, and draw a wire from your last point to the pin. It will also add a tap symbol called "CTAP" by default, between the bus you chose and the first wire segment. It will give the BN property on the pin of the CTAP symbol the bits you specified. It will add the signal name to the wire you are adding if you so desire (specify your choice with the set variable TAP_ADD_SIGNAL).

   Note: In this mode, you can terminate the tap wire, by clicking at the same point twice, (with the YELLOW (left) mouse button) after the tap symbol has been added. This is useful if you haven't yet added the component you want the tapped wire to go to, or you want the tapped wire to go to another wire etc.
4. The next signal name you entered, will be used for the next tap (i.e. go back to step 1 or 2).

Advanced User Section

Restrictions: The tap command works with very specific kinds of tap bodies.

Rules for creating a tap symbol are

The tap symbol should

- Have exactly two pins.
- One pin MUST be at the origin of the symbol.
- This pin may NOT have a BN property attached to it.

- The second pin MUST be located on the positive x axis i.e. its coordinates should be (x, 0), where x > 0.
- The second pin MUST be on a grid point.
- The second pin MUST have a BN property.

It is recommended that the first pin have a \NAC \NWC on it and the second pin have a \NAC.

As a default the tap symbol CTAP from the standard library is used. If you want to change the tap symbol, use the command

```
set TAP_BODY <yourtapname>
```

For example,

```
set TAP_BODY ktap
```

Related Commands

Bustap

Textsize

Syntax

```
textsize <size in inches> <group name>
```

to change text size of properties in a group.

OR

```
textsize <size in inches> Click on property
```

to change text size of a property.

```
textsize <size in inches> Click on note
```

to change the text size of a note.

You can specify a text size that has up to three decimal places. The minimum text size that you can specify is 0.008 inches and the maximum is 1.740 inches. The text size you specify should be a multiple of 0.002 inches.

Note: The above commands change the size of text in the plot.

Undo

Syntax

```
UNDo <cr>
```

Description

This command undoes the operation of the previous drawing command. The editor keeps a list of operations performed during the current editing session. Repeated applications of UNDO reverses the effects of events according to this list.

You can undo past a write to the beginning of a session if you edit a single drawing, write it, and then continue to edit the original drawing. If you edit a second drawing immediately after writing the first, and then return to the first drawing, you cannot undo past the point where you accessed the second drawing.

UNDO only affects the current drawing. For example, to undo the move command used to transfer an object from one drawing to another, type UNDO in the destination drawing to remove the object, and then type UNDO in the original drawing to replace the object.

Related Commands

Redo

Unhighlight

Syntax

```
UNHighlight [ Net | PArt | PIn | Any ] pt
UNHighlight [ Net | PArt | PIn | Any ] object_name
```

Net, Part, and Pin are used to specify the object type you want to unhighlight. You may use Any if you want all selected objects to be unhighlighted. This is also the default argument if you do not specify any object type.

pt To pick a component to be unhighlighted, point to the object and press the left button. Select a group by pointing to the required group and pressing the middle button.

object_name If you prefer typing in the name of the object, you may use the second version of this command.

For nets, the object name is the signal name of the net, for example, FOO.

For parts, the object name is the value of the PATH property attached to that part or component, for example, 7P.

For pins, the object name is the value of the PATH property attached to that part or component to which the pin belongs, followed by a period ("."), followed by the name of the pin, for example, 7P.A<SIZE-1..0>.

Wildcards such as * and ? may be used in specifying the object name,

for example, FOO*, 7*P, 7P.A*.

Description

The Unhighlight command is used to unhighlight an already highlighted object throughout the system.

Related Commands

Highlight
Dehighlight

Unix

Syntax

```
UNIx {operating_system_command  |  <cr> }
```

Description

This command accesses the operating system. To execute a particular system command, enter it on the same line as the SYSTEM command. Without an argument, the editor provides an interactive shell. You are connected to the operating system and can run any operating system commands.

To exit from the operating system and return to the editor, type Ctrl-D or Exit on UNIX-based systems.

Updatesheetvars

Syntax

```
updatesheetvars
```

Description

This command updates the custom text variables for page numbers on all pages in a design. For more information, see Updating Custom Text Variables for Page Numbers on page 363.

Use

Syntax

```
Use {library_name | <cr>}
```

Description

This command specifies a working library. Library_name refers to the name of the library you want to use. If the library is not in the current directory, include the pathname.

USE places the specified library at the top of the active search list, and it becomes your current working library. If the library has been previously specified, it is moved to the top of the library search stack.

There is no limit to the number of libraries that can be in use at one time.

USE <cr> brings up the file browser form. The user can then select a library_name from the form.

Related Commands

Directory
IGnore
Library

Vectorize

Syntax

```
VECtorize
```

Description

Note: This command works only on Solaris.

This command creates a file named `vector.dat`, which contains a vector plot format version of the current drawing. This file can be used to transmit files to other machines or drive a pen plotter (with the aid of a format conversion program).

Version

Syntax

```
VERsion {point | group_name}...
```

Description

This command selects an alternate version of a symbol. Some bodies are created with different symbolic representations. For example, the NAND gate is equivalent to an INVERT-OR gate by DeMorgan's Theorem. A NOR gate is equivalent to an INVERT-AND gate. All versions of a symbol refer to the same logic drawing.

To step from one representation of a symbol to another, type VERSION and point to a symbol. The editor determines the current version of the symbol and displays the next version in the sequence. Continue pressing the mouse button to cycle through all the symbol versions. After the last version of the sequence is displayed, the first version is redisplayed.

You can use the FIND command to group all occurrences of a specified symbol, and then issue the VERSION command with the group_name option to globally change the drawing. The center button changes the version of the bodies in the group closest to the cursor.

The separate versions of a symbol must all make reference to the same logic drawing. Using a different version of a symbol has no influence on the logic drawing defining it.

Related Commands

Add
Replace
Rotate

Vpadd

Syntax

```
VPAdd     <cr>
```

Description

This command creates a new viewport. Viewports let you look at different views of the same drawing or open different drawings at the same time. You can zoom the windows independently to focus on different sections of the design, use the WIRE and ROUTE commands to connect points between viewports, and use the new viewport as a global view of the original design.

When you create a viewport, the current drawing is copied to the new viewport and fit to the size of the window. Any operations you perform in either window appear on both copies of the drawing.

Select the active viewport by placing the cursor within the viewport. If the SET command option CLICK_TO_TYPE is ON, click the left mouse button within the window in order to make the window active.

To edit a different drawing, make sure the cursor is in the correct viewport and use the EDIT command to access the new drawing. When you edit different designs simultaneously, you can use the COPY and MOVE commands to share information between the drawings.

No matter which viewport is active, all commands and system responses appear in the message window of viewport1 (the main viewport). The message window is not considered a part of viewport1. If you move the cursor from one viewport into the message window of viewport1, the system still considers the viewport you were in as the active viewport. To activate viewport1, make sure the cursor enters the graphic window area of the viewport.

To move a viewport, place the cursor in the title bar of the viewport, press and hold the left mouse button, and move the viewport to the new location. To resize a viewport, place the cursor in a corner of the viewport, press and hold the left mouse button, and resize the viewport.

Vpdelete

Syntax

```
VPDelete point...
```

Description

This command deletes an existing viewport. VPDELETE remains active until you enter a semicolon or select another command.

To delete a viewport, click the left mouse button in the viewport. Any drawings that are active in the viewport are not saved but are noted as being modified drawings. Use the SHOW MODIFIED command to list drawings that have been changed but not written. To save the modified drawings, edit the drawing in another viewport and issue the WRITE command.

Window

Syntax

```
WINdow {; | Down | Fit | In | Left | Out | Previous |      point;|point point; |
point point point |Right  |  scale_factor |Up}...
```

Description

This command changes the view of the current drawing. WINDOW can use up to three arguments. If there are fewer than three arguments, terminate the command with a semicolon.

The command options are

WINDOW	Redraws the image without changing the center or scale. This refreshes the screen and clears all messages.
DOWN	Repositions the center of the screen down below the drawing (moves the drawing up on the screen).
FIT	Fits the drawing to the entire screen.
IN	Enlarges the size of the drawing on the screen.
LEFT	Repositions the center of the screen to the left of the drawing (moves the drawing right on the screen).

OUT	Reduces the size of the drawing on the screen.
PREVIOUS	Switches from the current window scale and positions to the previous window scale and position.
Pt ;	Pans the drawing and makes the point the center of a new screen display of the drawing. The scaling of the drawing remains the same. Use the right button to enter the single point.
Pt pt ;	Defines a rectangle with the specified points at opposite corners. The rectangle expands to fill the screen, providing a close-up view of the specified portion of the drawing.
Pt pt pt:	Changes the size of the drawing depending on the ratio between the points. The first point defines the new center of the drawing, and the display becomes either larger or smaller, depending on the other points. The drawing is redisplayed with the first point at the center. If the distance between the first point and the third point is greater than the distance between the first point and the second point, the items appear larger; if the distance is smaller, items appear smaller.
RIGHT	Repositions the center of the screen to the right of the drawing (move the drawing left on the screen).
Scale_factor	Enlarges or reduces the drawing by the specified factor. The center of the window remains the same.
UP	Repositions the center of the screen above the drawing (move the drawing down on the screen).

Related Commands

Zoom

Wire

Syntax

```
WIRe {[signal_name] point point}...
```

Description

This command adds wires to a drawing. The wire begins at the first point specified and runs to the second. Specify additional points to draw a wire with more segments. To snap the wire to the nearest vertex, press the right button.

To end a wire at a pin, dot, or other wire, press the left button.

To end a wire in a free space, press the left button twice at the final point.

Because schematics almost exclusively use orthogonal wires, the default wire mode is orthogonal (bent). Once the wire is started and the cursor changes direction, the attached wire remains orthogonal, whether the cursor is moved horizontally, vertically, or diagonally. To bend a wire, press the left button. Press the center button to change the orientation of the bend. If you press the center button a second time, the wire becomes diagonal. A third press returns the wire to the first orthogonal position.

If you enter a signal_name, the wire will be given that signal name.

See the BWIRE, BROUTE, ROUTE commands. See also the SET and SHOW commands to change default wiring behavior.

Related Commands

Broute
Bwire
Route

Write

Syntax

```
WRite [<directory>][drawing_name][.[type][.[version][.[page]]]] | <cr>
```

Description

This command writes the current drawing onto the disk. WRITE always asks for confirmation to write a drawing, even if no errors exist in the drawing. This is a safety feature to prevent unintentional writes. <DIRECTORY> is the directory where the drawing resides. If no directory is given, the drawing is written to the directory from which it was retrieved. If you write a newly-created drawing without giving a directory name, the drawing is written to the current directory. If only a directory name is given, the drawing is written into the specified directory. This is useful when copying drawings between directories.

DRAWING_NAME is the name of the drawing to write. If no drawing_name is specified (WRITE <cr>), the drawing is written to the file name shown on the status line at the top of the display. If no drawing type, version number, or page number are specified, the default is SCH.1.1.

If you enter a drawing name and a drawing with that name already exists in a directory, a warning message is displayed. Type YES to overwrite the existing drawing with the new drawing. Type NO to cancel the write.

Related Commands

Diagram
Edit
Get
set HDL_Direct

Zoom

Syntax

```
Zoom{;| Down |Fit |In | Left | Out | Previous | point; | point point; | point point
point | Right |  scale_factor | Up }...
```

Description

This command reduces and enlarges portions of the drawing on the screen. ZOOM can use up to three arguments. If there are fewer than three arguments, terminate the command with a semicolon. The command options are:

ZOOM	Redraws the image without changing the center or scale. This refreshes the screen and clears all messages.
FIT	Fits the drawing to the entire screen.
LEFT	Repositions the center of the screen to the left of the drawing (moves the drawing right on the screen).
PREVIOUS	Switches from the current window scale and position to the previous window scale and position.
Pt pt	Defines a rectangle with the specified points at opposite corners. The rectangle expands to fill the screen, providing a close view of the specified portion of the drawing.
RIGHT	Repositions the center of the screen to the right of the drawing (moves the drawing left on the screen).
U	Repositions the center of the screen up above the drawing (moves the drawing down on the screen).

Scale_factor	Enlarges or reduces the drawing by the specified amount. The center of the window remains the same.
Pt pt	Changes the size of the drawing depending on the ratio between the points. The first point defines the new center of the drawing, and the display becomes either larger or smaller, depending on the other points. The drawing is redisplayed with the first point at the center. If the distance between the first point and the third point is greater than the distance between the first point and the second point, the items appear larger. If the distance is smaller, items appear smaller.
Pt	Pans the drawing and make that point the center of a new screen display of the drawing. The scaling of the drawing remains the same. Use the right button to enter the single point.
OUT	Reduces the size of the drawing on the screen.
IN	Enlarges the size of the drawing on the screen.
DOWN	Repositions the center of the screen down below the drawing (moves the drawing up on the screen).

Related Commands

Window

B

Nongraphical Concept HDL (nconcepthdl)

The graphical Concept HDL:

- Only runs on graphics workstations
- Always draws graphics on the screen
- Only runs in the background under the X-Windows system

`nconcepthdl` is a version of Concept HDL that allows you to run Concept HDL scripts in a nongraphical mode. This allows you to run Concept HDL

- Without a graphics terminal
- In the background

`nconcepthdl` is useful for running a large batch processes, such as `hardcopy` or `backannotate`, without having to invoke Concept HDL.

Running nconcepthdl

To run `nconcepthdl`, use the following syntax:

```
nconcepthdl
     -proj <project file>.cpm
     [-scr <additional script filepath relative to the project directory>]
     [-log <log filepath>]
     [-ignoreprojscr]
```

`nconcepthdl` first executes the Concept HDL script specified in the `<project file>.cpm` file, if the script exists. Then if the `-scr` option is also specified, that script is executed. If you use the `-ignoreprojscr` option, the Concept HDL script specified in the `<project file>.cpm` file will not be executed.

If the `-log` option is not specified, the messages are thrown to `stderr` as well as the `nconcept.log` file in the project's temporary directory.

`nconcepthdl` supports all Concept HDL console commands except the Windows plotting command (`plot`). For more information on the Concept HDL console commands, see Appendix A, "Console Command Reference."

C

Using the Standard Library Symbols

The Standard Library is a Concept HDL library of symbols that are useful for manipulating signals in a structured design, applying properties to an entire design, and documenting information on the schematic. All the symbols in the Standard Library are supported by HDL Direct and are translated correctly into VHDL and Verilog text descriptions.

The Standard library is included in the default list of project libraries for all projects.

VHDL_DECS and VERILOG_DECS Symbols

The Standard Library includes two declaration symbols, VHDL_DECS and VERILOG_DECS. These symbols are not required on schematics, but they are useful for adding Verilog and VHDL-related properties that are applicable to the entire schematic. For example, to set the VHDL logic type for all vectored ports and signals in your drawing, you can add a VHDL_DECS symbol to the first page of the schematic and attach a VHDL_VECTOR_TYPE property to it.

Use the VHDL_DECS symbol for VHDL designs and the VERILOG_DECS symbol for Verilog designs. The two symbols are similar.

Follow these rules if you use a VHDL_DECS or VERILOG_DECS symbol:

- Use either symbol, VHDL_DECS or VERILOG_DECS, but not both, on a schematic.
- Place the symbol on the first page of the schematic.

Properties on VHDL_DECS and VERILOG_DECS

You can use the following properties on a VHDL_DECS or VERILOG_DECS symbol. The default values of all these properties, except VHDL_GENERICxx, VLOG_PARAM, and /PARAM, are specified in the *Output* tab of the *Concept Options* dialog box and apply to the entire design. By using these properties on a VHDL_DECS or VERILOG_DECS symbol, you can override the defaults for individual drawings.

Properties used when only VHDL is generated from a Concept HDL design

- VHDL_GENERICxx
- VHDL_SCALAR_TYPE
- VHDL_VECTOR_TYPE
- LIBRARYn
- USEn

Properties used when only Verilog is generated from a Concept HDL design

- VLOG_PARAMxx
- \PARAM
- VLOG_NET_TYPE

Customizing the VHDL_DECS or VERILOG_DECS Symbol

If the VHDL_DECS and VERILOG_DECS symbols are not suitable for your project, you can customize them by either creating a new symbol or by editing a copy of the symbol. You can also use a page border as a declarations symbol.

To create a new declarations symbol

1. Create a new symbol.
2. Copy all visible and invisible properties from the VHDL_DECS or VERILOG_DECS symbol in the Standard Library to the new declarations symbol and define their values.

To edit a VHDL_DECS or VERILOG_DECS symbol

1. Copy the VHDL_DECS or VERILOG_DECS symbol to a new cell.
2. Edit the symbol.

Note: The declarations symbol must have the HDL_SCHEMATIC = TYPE1 property. Do not change its value.

Using a Page Border as a Declarations Symbol

If you do not want to add a VHDL_DECS or VERILOG_DECS declarations symbol to your schematic, you can use a page border that functions as a declarations symbol.

To use a page border as a declarations symbol

1. Add a page border from the Standard Library to your schematic.
2. Remove the COMMENT_BODY = TRUE property from the page border.
3. Copy all visible and invisible properties from a VHDL_DECS or VERILOG_DECS symbol to the page border and define their new values.

Note: Because VHDL_DECS or VERILOG_DECS properties can appear on only the first page of a multiple-page schematic, you must use a different page border for the other pages. The page border for the other pages must have the COMMENT_BODY = TRUE property.

SYNOP_DEC Symbol

Do not use the SYNOP_DEC symbol, an old symbol still included in the Standard Library. Use a VHDL_DECS or VERILOG_DECS symbol instead. For more information, see VHDL_DECS and VERILOG_DECS Symbols on page 613.

The SYNOP_DEC symbol was used for compatibility with Synopsys tools. It is similar to the DECLARATIONS symbol in the Standard library, except for the following:

- The SYNOP_DEC symbol does not have the USER = WORK.ALL property.
- The SYNOP_DEC symbol includes a SYNOPSYS_PRAGMA property to support non-integer generic parameters. Because Synopsys tools do not support generic parameters that are not integers, HDL Direct adds the following lines around the information about non-integer generic parameters in the entity and architecture files:

```
pragma translate_off
pragma translate_on
```

Note: If you use the SYNOP_DEC symbol, the VHDL netlist generated by Concept HDL does not adhere strictly to the VHDL language guidelines. While elaborating the design, use the Leapfrog compatibility option (-c) so that the Leapfrog simulator will accept non-conforming VHDL syntax.

DECLARATIONS Symbol

Do not use the DECLARATIONS symbol, an old symbol still included in the Standard library. Use a VHDL_DECS or VERILOG_DECS symbol instead. For more information, see VHDL_DECS and VERILOG_DECS Symbols on page 613.

HDL_DECS Symbol

Do not use the HDL_DECS symbol, an old symbol still included in the Standard Library. Use a VHDL_DECS or VERILOG_DECS symbol instead. For more information, see VHDL_DECS and VERILOG_DECS Symbols on page 613.

TAP Symbols

Use tap symbols to “tap” or extract a single bit or a range of bits from a vectored signal (bus).

The Standard library has the following tap symbols:

- TAP
- CTAP
- BIT TAP
- LSBTAP
- MSBTAP

All these symbols, except CTAP, have many versions, depending on the rotation of the symbol.

To use a TAP symbol, do one of the following:

- Use the *Component > Add* command to add one of the tap symbols contained in the Standard library. Select the version of the symbol while adding it or after you place it on the schematic.
- Use the *Wire > Bus Tap* command. You can choose the default tap symbol for the *Wire > Bus Tap* command by specifying it in *Graphics* tab of the *Concept Options* dialog box.

TAP

Use a TAP symbol to tap or extract a single bit from a bus. The BN property on the TAP symbol determines which bit is tapped. Set the value of this property to the actual bit number you want to tap. The BN property does not have a default value; you must specify its value.

Note: TAP symbols are similar to CTAP symbols. The only difference between them is their graphical representation.

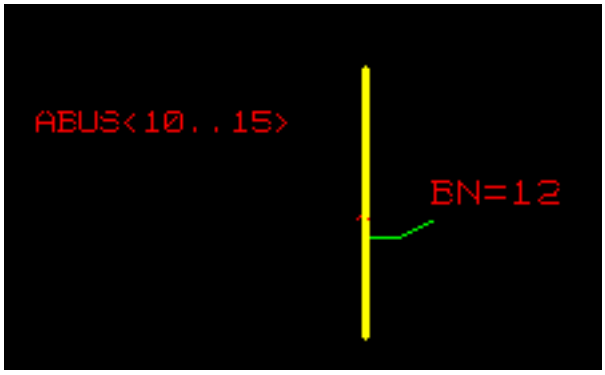
To tap a bit with a TAP symbol

1. Attach a TAP symbol to the bus.
2. Use the *Text > Change* command to set the value of the BN property to the bit number that you want to tap.

Example

To select the twelfth bit of the bus ABUS<10..15>,

1. Attach a TAP symbol to the bus.
2. Set the value of the BN property on the TAP symbol to 12.



Guidelines for Creating Tap Symbols

Apply these rules to any tap symbols that you create:

1. The tap symbol must have exactly two pins.
2. One pin *must* be at the origin of the tap symbol.
3. The second pin must be on a grid point.

4. The second pin must be located on the x-axis, and x must be > 1.

Attach a BN property to the second pin (*not* to the first pin).

CTAP

Use a CTAP symbol to “tap” or extract a single bit from a bus. The BN property on the CTAP symbol determines which bit is tapped. Set the value of this property to the actual bit number you want to tap. The BN property does not have a default value; you must specify its value.

Note: CTAP symbols are similar to TAP symbols. The only difference between them is their graphical representation.

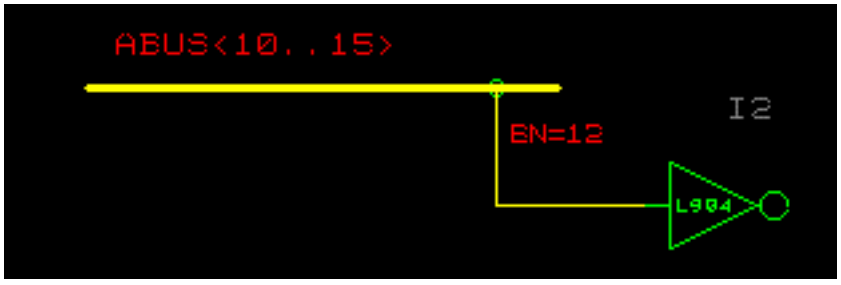
To tap a bit with a CTAP symbol

1. Attach a CTAP symbol to the bus.
2. Use the *Text > Change* command to set the value of the BN property to the actual bit number that you want to tap.

Example

To select the twelfth bit of the bus ABUS<10..15>,

1. Attach a CTAP symbol to the bus.
2. Set the value of the BN property on the CTAP symbol to 12.



BIT TAP

Use the BIT TAP symbol to “tap” or extract a single bit from a bus. The BIT property on the BIT TAP symbol determines which bit is tapped. The value of this property is relative, beginning from the Least Significant Bit (LSB). For example, for a bus addr<10..15>, if BIT

= 3 on the BIT TAP symbol the twelfth bit of the bus is tapped. The default value of the BIT property on BIT TAP symbols is 0.

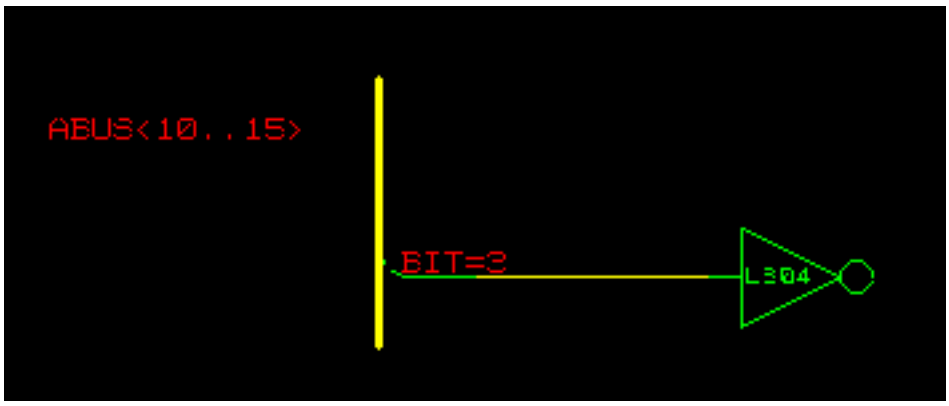
To tap a bit with a BIT TAP symbol

1. Attach a BIT TAP symbol to the bus.
2. Use the *Text* > *Change* command to set the value of the BIT property on the symbol to the bit you want to tap. This value must be between 0 and <bus_size>-1.

Example

To select the twelfth bit of the bus ABUS<10..15>,

1. Attach a BIT TAP property to the bus.
2. Set the value of the BIT property on the BIT TAP symbol to 3.



LSBTAP Symbol

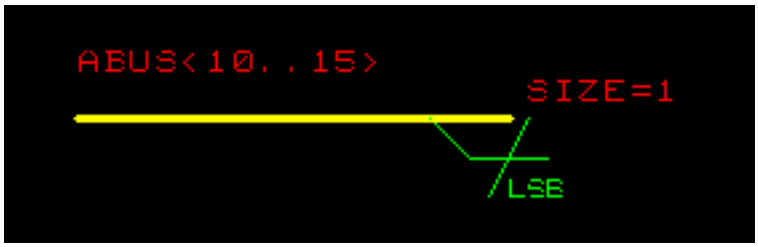
Use the LSBTAP symbol to “tap” or select the Least Significant Bit (LSB) of a bus, or a range of bits beginning with the LSB. The SIZE property on the LSBTAP symbol determines which bits are tapped. The default value of this property is 1, which means that the LSB is tapped.

To tap a single bit

➤ Attach an LSBTAP symbol to the bus. The LSB is tapped.

Example: Using LSBTAP to tap a single bit

To tap the LSB of a bus ABUS<10..15>, attach an LSBTAP symbol to the bus. The default value of the SIZE property on the LSBTAP symbol is 1. The tenth bit of the bus is tapped.



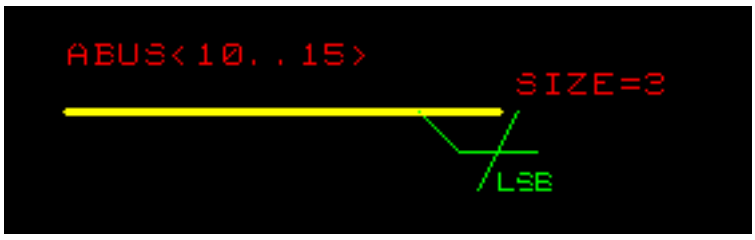
To tap a range of bits

1. Attach an LSBTAP symbol to the bus.
2. Use the *Text > Change* command to set the value of the SIZE property to the number of bits you want to tap.

Example: Using LSBTAP to tap a range of bits

To tap bits 10, 11, and 12 from a bus ABUS<10..15>,

1. Attach an LSBTAP symbol to the bus.
2. Set the value of the SIZE property on the LSBTAP symbol to 3.



MSBTAP Symbol

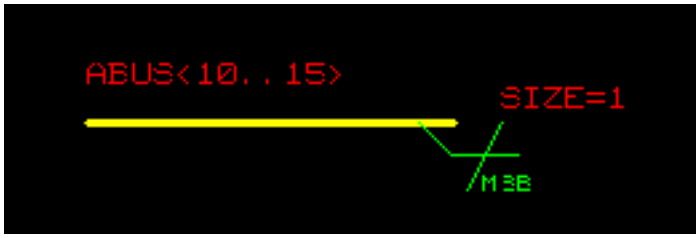
Use the MSBTAP symbol to “tap” or select the Most Significant Bit (MSB) of a bus, or a range of bits beginning with the MSB. The SIZE property on the MSBTAP symbol determines which bits are tapped. The default value of this property is 1, which means the MSB is tapped.

To tap a single bit

➤ Attach an MSBTAP symbol to the bus. The MSB is tapped.

Example: Using MSBTAP to tap a single bit

To tap the MSB of bus ABUS<10..15>, attach an MSBTAP symbol to the bus. The default value of the SIZE property on the MSBTAP symbol is 1. The fifteenth bit is tapped.



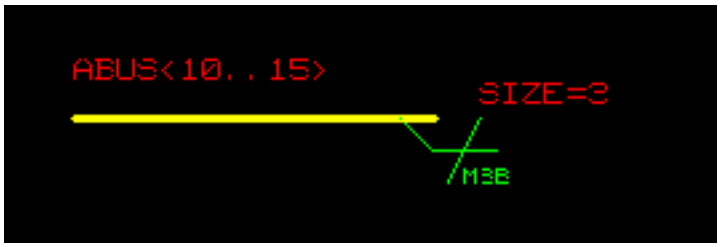
To tap a range of bits

1. Attach an MSBTAP symbol to the bus.
2. Use the *Text* > *Change* command to set the value of the SIZE property to the number of bits you want to tap.

Example: Using MSBTAP to tap a range of bits

To tap bits 13, 14, and 15 from a bus ABUS<10..15>,

1. Attach an MSBTAP symbol to the bus.
2. Set the value of the SIZE property on the MSBTAP symbol to 3.



CONCAT Symbols

Use a CONCAT symbol when you want to merge a number of signals, ports, or signal aliases into a group. You can then route this group to a port or instance with a single wire. For more

information on using CONCAT symbols to concatenate signals, ports, or signal aliases, see Signal Concatenations on page 147.

There are nine CONCAT symbols in the Standard library: CONCAT2, CONCAT3, CONCAT4, CONCAT5, CONCAT6, CONCAT7, CONCAT8, CONCAT9, CONCAT10. Use CONCAT2 to merge two signals, CONCAT3 to merge three signals, CONCAT4 to merge four signals, and so on. Each CONCAT symbol has a small note at the top of the symbol indicating the left pin and a small note at the bottom of the symbol indicating the right pin. You will have to zoom in to see these notes.

Concatenated signals can be separated back into individual signals with a MERGE or TAP symbol. (Concept HDL works faster if you use one of the TAP symbols instead of a MERGE to slice signals.)

Concatenated symbols are unrelated; concatenation is merely a shorthand notation for signals that run together.

Rules for Using CONCAT Symbols

- The sum of the input signal width must match the width of the output signal connected to the CONCAT symbol. If the input signal or output signal is not named, Concept HDL calculates the signal width automatically.
- If the widths of the input signals are specified, and the output signal is not named, the width of the output signal is assumed to be the sum of the widths of the input signals.
- The output of a concatenation can be connected to the input of a CONCAT symbol.
- The output of a slice can be connected to the input of a CONCAT symbol.
- When you create or edit a CONCAT symbol ensure that:
  - The name of the output pin is the highest alphanumeric value of all the pins on the symbol. For example, if the input pins are named AA, DD, and FF, the output pin cannot be named BB.
  - The VHDL_CONCAT=TRUE property is attached to the CONCAT symbol.

SYNONYM

The SYNONYM symbol is used to specify another name for a signal. SYNONYM symbols are useful for creating locally meaningful names for signals that are spread throughout a design.

Do not use SYNONYM symbols if you want to generate VHDL text for the schematic. Use ALIAS symbols instead. SYNONYM symbols are similar to ALIAS symbols, but Concept HDL does not have extensive VHDL checks for SYNONYM symbols. If you use SYNONYM symbols, Concept HDL will not detect all the VHDL-related errors and your VHDL output will be inaccurate. Therefore, if you currently have SYNONYM symbols in a design for which you will generate VHDL text, replace them with ALIAS symbols. If you do not intend to generate VHDL text from your schematic, you can use either ALIAS symbols or SYNONYM symbols. For more information on the ALIAS symbol, see Creating an Alias for a Signal on page 137.

To create a SYNONYM for a signal

1. In Concept HDL, choose *Component > Add*.

   The *Component Browser* appears.

2. In the *Library* drop-down, select *Standard*.

3. In the *Cells List*, select *SYNONYM*.

   The SYNONYM symbol gets attached to the cursor.

4. Click in the Concept HDL drawing area to place the symbol.

5. Attach the signal for which you want to create a synonym to the left pin of the SYNONYM symbol.

6. Attach the SYNONYM name to the right pin.

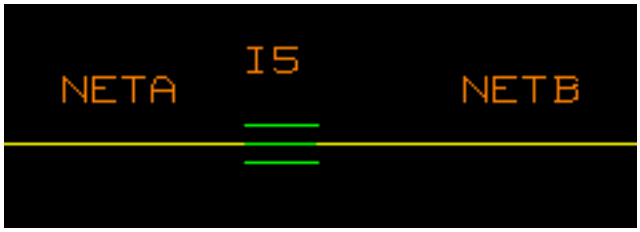
Example

To create a synonym `NETB` for `NETA`,

1. Add a SYNONYM symbol from the Standard Library.

2. Attach a wire with the signal name `NETA` to the left pin of the SYNONYM symbol.

3. Attach a wire to the right pin of the SYNONYM symbol and name it NETB.



This creates the following Verilog declaration:

```
alias_bit alias_inst1 (netb, neta);
```

Rules for Using SYNONYM Symbols

You must follow the following rules for using the SYNONYM symbols:

- Connect signals of the same assertion: both must be high or both must be low.
- Connect signals of the same width.
- Synonym symbols cannot be used if you want to generate VHDL text for the schematic. Use ALIAS symbols instead. For more information on the ALIAS symbol, see Creating an Alias for a Signal on page 137.

PAGE Borders

The Standard Library includes several page borders that you can use in your schematic. These provide a convenient way of documenting information such as the date, the design name, the page number, the engineer's name, and the company logo on the schematic.

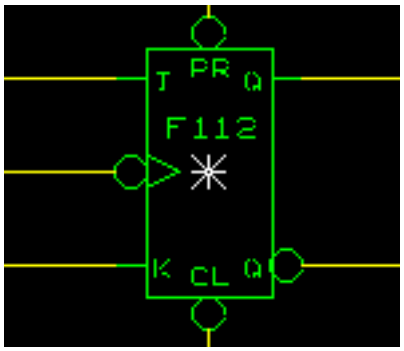
The following page borders are in the Standard Library:

A Size Page	The A SIZE PAGE symbol is an 8 1/2 x 11 inch border.
B Size Page	The B SIZE PAGE symbol is a 11 x 17 inch border.
C Size Page	The C SIZE PAGE symbol is a 17 x 22 inch border.
D Size Page	The D SIZE PAGE symbol is a 22 x 34 inch border.
E Size Page	The E SIZE PAGE symbol is a 34 x 44 inch border.
F Size Page	The F SIZE PAGE symbol is a 44 x 68 inch border.

ORIGIN

The ORIGIN symbol is a pseudo symbol that is required in all symbols. It provides an attachment location for symbol properties and defines the “origin” of the symbol. It is a pseudo symbol because it is the only symbol allowed within another symbol. It is not instantiated and, therefore, not visible in a schematic.

Concept HDL automatically uses the ORIGIN symbol to indicate the origin of any symbol. You do not add this symbol manually to a drawing. When you create a new symbol drawing, the ORIGIN symbol appears in the center of the screen.



To view the ORIGIN symbol

➤ In Concept HDL, choose *Display* > *Origins*.

DRAWING

Use DRAWING symbols to attach properties to all instances in the schematic.

If the drawing symbol is instantiated in the schematic and the property ABC=EFG is attached to it, all instances within the schematic will get this property. If any particular instance has this property with a different value, then the new value is applicable for the instance.

By default, the DRAWING symbol has the LAST_MODIFIED property. The value of the property is the date and time the schematic was last updated. This is set automatically.

Examples of other properties that can be attached to the DRAWING body are the TITLE and ABBREV properties. The TITLE property specifies the title of the drawing and must match the schematic name. The ABBREV property specifies an abbreviation of the drawing name.

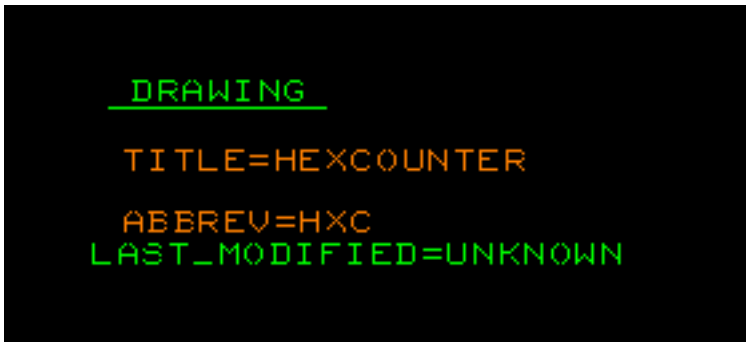
To use a DRAWING symbol

1. Add the symbol to the schematic.
2. Attach the properties to the symbol.

Note: Text macros cannot be used in the value of a `TITLE` property.

Example

The following drawing symbol has the `TITLE` and `ABBREV` properties in addition to the `LAST_MODIFIED` property.

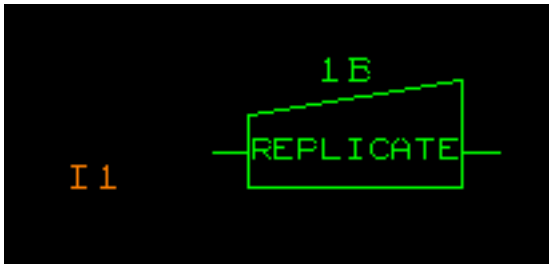


REPLICATE

REPLICATE symbols are usually not added to schematics. They are used by library developers to make models for sizable parts.

Note: You cannot connect a vector net to the input of a REPLICATE symbol.

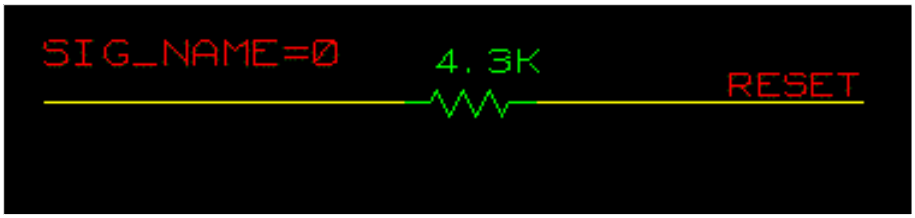
Example



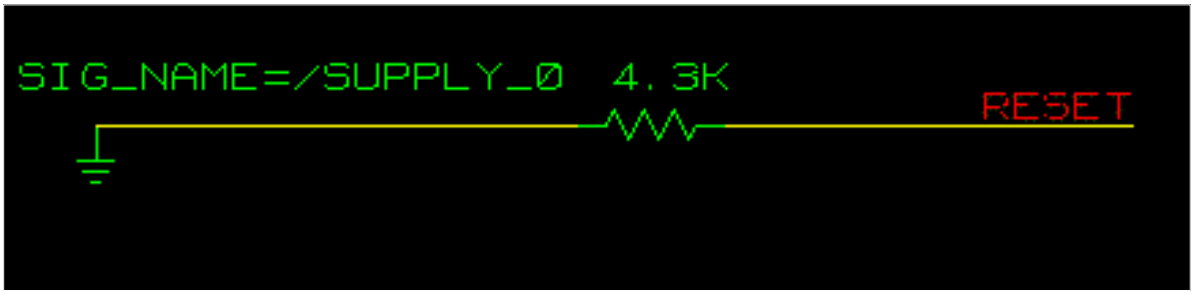
SUPPLY_0

Signals can be defined for the supply0 type in the following ways:

- Use the SUPPLY_0 symbol. The SUPPLY_0 symbol represents a global power signal. You can use this symbol instead of the SUPPLY_0 signal name. In this case, signal supply_0 is of the type supply0.



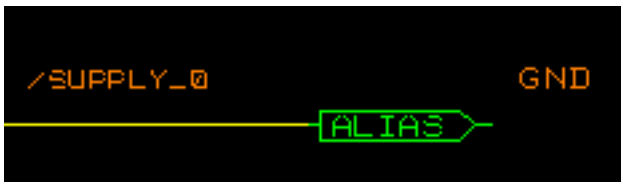
- Use the `/SUPPLY_0` signal name on any wire. In this case, the signal supply_0 is of the type supply0.



- Name the wire `0`. (Do not put a / in front of 0). In this case, signal 0 will be of type supply0.

- Use the ALIAS symbol to rename the `/SUPPLY_0` signal in the first page of the schematic and then use the alias in other pages. In Figure C-1, the ground signal `\SUPPLY_0` is given the alias GND. You can use the signal name GND in other areas of the schematic to represent a ground signal.

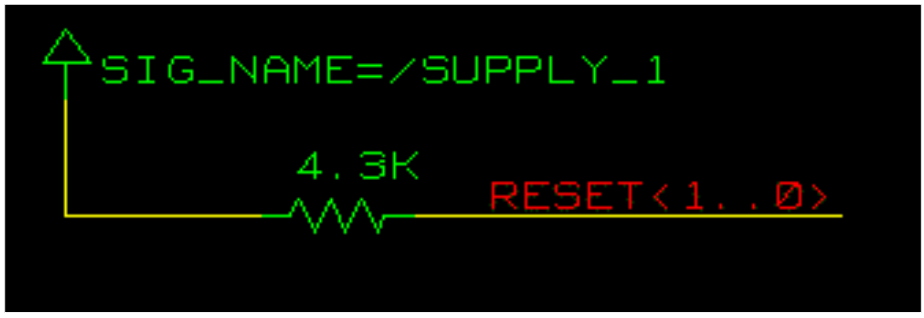
Figure C-1 Renaming a Global Ground Signal



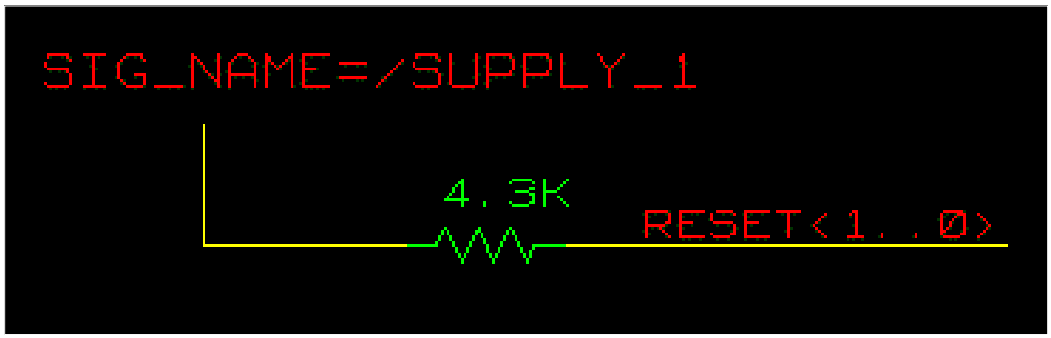
SUPPLY_1

Signals can be defined for the supply1 type in the following ways:

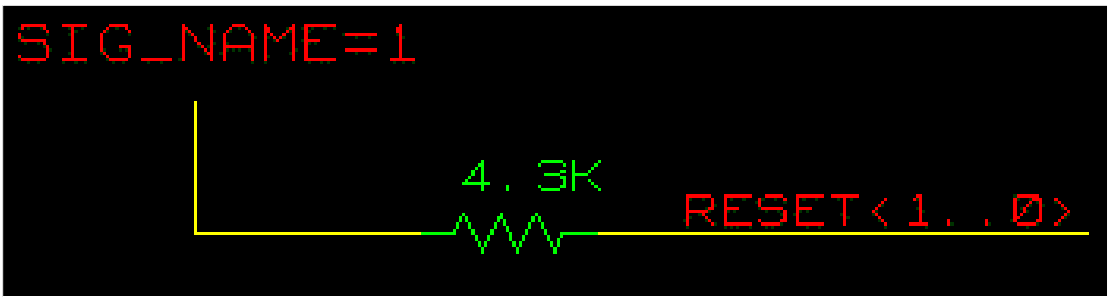
- Use the SUPPLY_1 symbol. The SUPPLY_1 symbol represents a global power signal. You can use this symbol instead of the SUPPLY_1 signal name. In this case, the signal supply_1 is of the type supply1.



- Use the /SUPPLY_1 signal name on any wire. In this case, signal “supply_1” will be of the type supply1.



- Name the wire 1. (Do not put a / in front of 1). In this case, signal 1 will be of the type supply1.



- Use the ALIAS symbol to rename the /SUPPLY_1 signal in the first page of the schematic and then use the alias in other pages. In Figure C-2, the power signal \SUPPLY_1 is given the alias VCC. You can use the signal name VCC in other areas of the schematic to represent a power signal.

Figure C-2 Renaming a Global Power Signal

PIN NAMES

PIN NAMES symbols are used for hierarchical designs and library development. When you create a hierarchical schematic-symbol drawing pair, use the PIN NAMES symbol to transfer the `PIN_NAME` properties from the symbol drawing to its corresponding schematic drawing. Using the PIN_NAMES symbol eliminates the need to retype signal names and reduces errors in labeling signals and properties.

To use the PIN NAMES symbol

1. Create the symbol drawing.
2. Add pin names to the symbol with the *Wire > Signal Name* command. A `PIN_NAME` property is attached to each of the pins you name.
3. Save the symbol drawing.
4. Create the corresponding schematic drawing. The schematic drawing must have the same name as the symbol drawing, but with a `.sch` extension.

   For example, if the symbol drawing is `CLOCK.SYM.1.1`, type the following in the Concept HDL console window:

   ```
   edit clock
   ```

   The `CLOCK.SCH.1.1` drawing will contain the logic that the symbol represents. Place all the required parts and attach wires as required.
5. Add the PIN NAMES symbol from the `standard` library to a corner of the schematic.

   You can also add the PIN NAMES symbol on the schematic using the `pinnames` Concept HDL console window command. For more information, see The pinnames Command on page 520.
6. Run the `check` console window command.

   Concept HDL automatically attaches the names of the pins on the corresponding symbol drawing to the PIN NAMES symbol you added and appends a `\I` suffix (scope = interface) to each signal name. Each pin name is identified with a `SIG_NAME` property.

   Do one of the following if you want to view the signal and property names:

   - Place the cursor on a pin name attached to the PIN NAMES symbol
   - Choose *Text > Attributes* and click on the PIN NAMES symbol to view the signal and property names in the *Attributes* dialog box.

7. Choose *Text > Reattach* to reattach the individual signal names from the PIN NAMES symbol to the appropriate signals on the schematic drawing.
8. Choose *Display > Attachments* to ensure that the signal names have been reattached to the appropriate signals.
9. For drawing clarity, choose *Edit > Move* to relocate the signal names near the associated signals.
10. Delete the PIN NAMES symbol.

FLAG

FLAG symbols are attached to interface signals to indicate the physical pins of a design. FLAG symbols are usually not used; however, they are required as Packager-XL output by some physical design systems.

Note: The FLAG symbol is similar to the PORT symbols in the Standard library. Use PORT symbols instead of FLAG symbols. For more information on PORT symbols, see Adding Ports on page 132.

The FLAG symbol has 12 versions:

- Versions 1-4: Input
- Versions 5-8: Output
- Versions 9-12: In/Out

NOT

Do not use the NOT symbol, an old symbol still contained in the Standard library. The symbol was used only to support the Bubble Checker feature of the SCALD compiler, which verified that signals and pins were connected to signals and pins of the same assertion. The NOT symbol was used to avoid this restriction and provided a way of connecting signals and pins of the opposite assertion.

If you already have NOT symbols on your schematic, you do not have to remove them. They will be ignored and the two signals on either side of the NOT symbol will be synonymed together.

Note: An unnamed net cannot be used with a NOT symbol unless you specify the size of the net with a SLASH symbol.

DEFINE

Use the DEFINE symbol when you want to

- Use X-Replication

  For more information on using X-Replication, see X-Replication on page 168.

- Define Text Macros

  For more information on text macros, see the *Packager-XL Reference Guide*.

SIM_DIRECTIVES

Do not use the SIM_DIRECTIVES symbol, an old symbol that was used to pass simulator directives to RapidSIM.

Other Symbols

For information about	See
ALIAS	Creating an Alias for a Signal on page 137
IOPORT, INPORT, LNKPORT, BUFPORT, OUTPORT, AOUTPORT	Adding Ports on page 132
MERGE	Merge Symbols on page 150
SLASH	Specifying the Size of Nets on page 135
SLICE	Signal Slices (Bit and Part Selects) on page 154

Customizing Standard library Symbols

To customize a Standard library symbol,

- Copy the symbol to a new cell and edit it. You can change the shape of the symbol, but do not change its visible or invisible properties.

  OR

- Create a new symbol and copy the properties from the corresponding Standard library symbol to the new symbol.

D

Error Checking Features in Concept HDL

Concept HDL performs the following error checking functions when generating the netlist for a design:

- Cross-View Checking on page 633
- Entity Declaration Checking for Instantiated Components on page 636

You can specify additional error-checking options in the *Check* tab of the *Concept Options* dialog box.

To specify additional error checking options in Concept HDL

1. Choose *Tools* > *Options*.

   The *Concept Options* dialog box appears.

2. Select the *Check* tab and specify the error checking options.

   For more information on error-checking options, see the Concept HDL online help.

Cross-View Checking

Cross-view checking is an error-checking feature of ports, port modes, and port types that is performed between the schematic views and the symbol views.

When you save a schematic, Concept HDL compares the ports, port modes, and port types in the schematic views and the symbol views and checks if they are consistent. If there are any inconsistencies, Concept HDL displays an error or warning message.

- Concept HDL displays the following error message if all the ports present in the schematic views are not present in the symbol views:

  ```
  168 ERROR Schematic has port but port does not exist in the symbol. Either
  delete this port from the schematic or add this port in the symbol.
  ```

  To correct this error, do one of the following:

- Delete extra ports in the schematic views
- Add missing ports in the symbol views

- Concept HDL displays the following warning message if all the ports present in the symbol views are not present in the schematic views:

  ```
  171 WARNING Port exists in symbol but not in the schematic. Either delete this
  port from the symbol or add this port in the schematic.
  ```

  To correct this error, do one of the following:

  - Add missing ports in the schematic views
  - Delete extra ports in the symbol views

  If you have less ports in the schematic views than in the symbol views, Concept HDL displays a warning message because the additional ports found on the symbol are left open in the netlist.

  Note: In the *Markers* dialog box, if you click on the warning message that is displayed when ports that are present in the symbol views are not present in the schematic views, the additional ports on the symbol will not get highlighted in Concept HDL. However, you can view the port name and port type of the additional ports in the *Markers Detail* dialog box.

- Concept HDL displays the following error message if the port mode declared in the schematic view is different from the port mode declared in the symbol view:

  ```
  169 ERROR Port mode for the pin on the symbol is different from that on the pin
  of the instance. Modify the port mode to make it the same.
  ```

  To correct this error, modify the port mode of the symbol or the schematic view to make it the same. For more information, see Declaring Port Modes on page 183.

- Concept HDL displays the following error message if the port type declared in the schematic view is different from the port type declared in the symbol view:

  ```
  170 ERROR Port type specified in the schematic and symbol is different. Modify
  schematic/symbol to make port type same.
  ```

  To correct this error, modify the port type of the symbol or the schematic view to make it the same. For more information, see Setting the Verilog Logic Type for a Specific Port on page 159 and Setting the VHDL Logic Type for a Specific Port on page 162.

When you save a symbol, Concept HDL compares the ports, port modes, and port types in the symbol views and the schematic views and checks if they are consistent. If there are any inconsistencies, Concept HDL displays an error or warning message.

- Concept HDL displays the following error message if all the ports present in the schematic views are not present in the symbol views:

```
168 ERROR Schematic has port but port does not exist in the symbol. Either
delete this port from the schematic or add this port in the symbol.
```

To correct this error, do one of the following:

- Delete extra ports in the schematic views
- Add missing ports in the symbol views

- Concept HDL displays the following warning message if all the ports present in the symbol views are not present in the schematic views:

  ```
  171 WARNING Port exists in symbol but not in the schematic. Either delete this
  port from the symbol or add this port in the schematic.
  ```

  To correct this error, do one of the following:

  - Add missing ports in the schematic views
  - Delete extra ports in the symbol views

  If you have less ports in the schematic views than in the symbol views, Concept HDL displays a warning message because the additional ports found on the symbol are left open in the netlist.

  Note: In the *Markers* dialog box, if you click on the warning message that is displayed when ports that are present in the symbol views are not present in the schematic views, the additional ports on the symbol will not get highlighted in Concept HDL. However, you can view the port name and port type of the additional ports in the *Markers Detail* dialog box.

- Concept HDL displays the following error message if the port mode declared in the symbol view is different from the port mode declared in the schematic view:

  ```
  169 ERROR Port mode for the pin on the symbol is different from that on the pin
  of the instance. Modify the port mode to make it the same.
  ```

  To correct this error, modify the port mode of the symbol or the schematic view to make it the same. For more information, see Declaring Port Modes on page 183.

- Concept HDL displays the following error message if the port type declared in the symbol view is different from the port type declared in the schematic view:

  ```
  170 ERROR Port type specified in the schematic and symbol is different. Modify
  schematic/symbol to make port type same.
  ```

  To correct this error, modify the port type of the symbol or the schematic view to make it the same. For more information, see Setting the Verilog Logic Type for a Specific Port on page 159 and Setting the VHDL Logic Type for a Specific Port on page 162.

Entity Declaration Checking for Instantiated Components

When Concept HDL generates the VHDL netlist for a schematic, it also generates a VHDL component declaration for each component used in the schematic. To generate this component declaration, Concept HDL reads the entity declaration associated with the component. For example, if the part NAND2 from the /usr/libs/lsttl library is instantiated in a schematic, Concept HDL reads the following file:

```
/usr/libs/lsttl/nand2/entity/vhdl.vhd
```

When Concept HDL has to read the entity declaration, there could be different situations:

- If an entity declaration for the part is found, the declaration is used to construct a component declaration. If you add the VHDL_GENERICS property to an instance on the schematic and if the same property is not found in the entity or the symbol of the component, it is declared in the component declaration section of the component.
- If no entity declaration is found and the component is placed in the schematic and saved, the following warning message is displayed:

  ```
  177 WARNING: Entity declaration for part does not exist in library
  ```

  The VHDL architecture is created and the component declaration section in the VHDL architecture (sch_n/vhdl.vhd) in this case is constructed from the symbol view of the component (sym_n/symbol.css). Because the symbol view does not contain information about the port mode of the component, Concept HDL declares all the ports as inout in the component declaration section.
- If the entity declaration is not present for the instance and the property VHDL_GENERICS is attached to the instance, the VHDL architecture will declare these generics in the component section in VHDL architecture (sch_n/vhdl.vhd), which gets constructed from the symbol view of the component (sym_n/symbol.css). Because the symbol view does not contain information about the port mode of the component, Concept HDL declares all the ports as inout in the component declaration section.

Netlisting Errors

This section describes some select netlisting errors that appear in the *Markers* window. Each error is listed in the ascending order of the error numbers. The solution or workaround for the error is described.

Click on an error message for detailed information on the error message.

Error
ERROR 105: SUPPLY_1 and SUPPLY_0 signal names must begin with a / (forward slash).
ERROR 106: SUPPLY_1 and SUPPLY_0 signals cannot be vectored.
ERROR 110: Range direction needs to be changed to ‘TO’.
ERROR 111: Range direction needs to be changed to ’DOWNTO’.
VHDL_ERROR 112: Value of VHDL_PORT property is set improperly.
ERROR 113: Port is attached to an unnamed signal.
ERROR 114: Port is not connected.
VHDL_ERROR 118: Signal has two different VHDL_VECTOR_TYPE values.
VHDL_ERROR 119: Signal has two different VHDL_SCALAR_TYPE values.
ERROR 120: Signal is declared to be both a port and an alias.
WARNING 121: Port has two different port modes. Port being declared as inout.
WARNING 122: Signal is a global signal in one place but not in the other place. The signal will be treated as a global signal at both the places.
ERROR 123: Same alias is made to two different signals.
ERROR 124: Signal is declared to be both a scalar and a vector.
ERROR 126: Identifier is used as both a PATH value and a signal name.
WARNING 127: Identifier is used as both a component name and a signal name.
ERROR 129: A global signal cannot also be a port.
ERROR 131: You cannot tap from an unnamed signal.
ERROR 132: You cannot tap from an unconnected signal.
ERROR 136: The signal coming out of the concatenation symbol must be unnamed.
ERROR 137: Each pin on a concatenation symbol must be connected to a signal.
ERROR 144: Alias symbol has an unconnected pin.
ERROR 145: Pin on alias symbol has an unnamed signal attached.
ERROR 146: Signal coming out of an alias symbol is also a port.
ERROR 147: Signal coming out of an alias symbol cannot be a global signal.
ERROR 150: Signal connected to pin has incorrect width.

Error
ERROR 151: Entity declaration for instance declares a port that is not on the instance.
ERROR 152: Port on instance does not exist in entity declaration for instance.
ERROR 153: Port on instance is vectored but port in entity declaration for instance is not.
ERROR 154: Port on instance is scalar but port in entity declaration for instance is not.
ERROR 155: Range direction for port on instance conflicts with port in entity declaration for instance.
ERROR 156: Instance port and entity port modes are incompatible.
ERROR 158: Sizeable pin cannot be represented in Verilog because it is partly unconnected.
ERROR 164: Pin width is greater than attached signal width.
ERROR 165: Concatenated signal width must match pin width.
ERROR 166: Attached signal width is not an integer multiple of pin width.
ENTITY ERROR 169: Port mode for the pin on the symbol is different from that on the pin of the instance. Modify the port mode to make it the same."
ERROR 174: Output of tap is unconnected.
WARNING 177: Entity declaration for part does not exist in library.
ERROR 178: Port exists in the entity but not on the instantiated symbol. Please rewrite the necessary pages.
ERROR 179: Two signal names are attached to this net.
ERROR 181: Signals on both the sides of the MERGE/TIE symbol are connected to driver pins of other instances.
ERROR 182: Signal on one side of the MERGE/TIE symbol is a global signal and signal on the other side is connected to driver pin of another instance.
ERROR 183: Both sides of the MERGE/TIE symbol are connected to global signals.
WARNING 184: A OUTPORT symbol in the schematic must not be connected to MERGE/TIE symbol.
ERROR 185: One of the pins of MERGE/TIE symbol is unconnected.
ERROR 187: Signals attached to MERGE/TIE symbol have parameterized width.
ERROR 188: Signals attached to each side of MERGE/TIE symbol have different width.
ERROR 190: Signals on both the sides of the MERGE/TIE symbol are undriven.

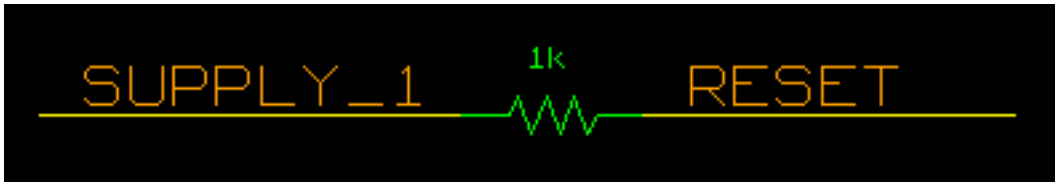
Error
WARNING 191: Cannot place pin properties on a pin with parameterized width if other pins on the instance have the same basename.
WARNING 192: Cannot place properties on specific bits of a signal which has parameterized width.
ERROR 197: Property on declarations symbol has incorrect value.
ERROR 198: Signal syntax is incorrect.
WARNING 211: Size Property not present on instance. Assuming a value of 1.
ERROR 212: Net connected to the output of Replicate instance should be unnamed or have the same width as the Replicate instance.
WARNING 217: Bit property not present on instance. Assuming a value of 0
ERROR 222: Error in symbol files.
ERROR 230: Net widths on both sides of the merge body do not match.
ERROR 231: Symbol pin is wider than the entity port.
ERROR 234: Different component uses same SPLIT_INST_NAME/SPLIT_INST prop value. Use different prop value for different components.
ERROR 260: Two assertion character - and * used in the signal name, it is not allowed. Use only one assertion character
ERROR 264: Property (SIZE/HAS_FIXED_SIZE/TIMES) can have only integer value. Ignoring this value and using 1 as the default value.
ENTITY_ERROR 267: Port range specified in the schematic and symbol is different. Modify schematic/symbol to make port range same.
ENTITY_ERROR 268: Port is specified vectored in the schematic but scalar on symbol. Modify schematic/symbol to make port consistent.
ENTITY_ERROR 269: Port is specified scalar in the schematic but vectored on symbol. Modify schematic/symbol to make port consistent.
ERROR 275: Two global signals are shorted.
WARNING 401: Binding Instance
ERROR 422: Chips File Packaging Error
ERROR 521: In Specifying Property On Instance
ERROR 526: In Specifying Split Inst Property on Instance

ERROR 105: SUPPLY_1 and SUPPLY_0 signal names must begin with a / (forward slash).

Description

This error has occurred because of the following reasons:

- The SIG_NAME property on the pin of the symbol for the component has the value SUPPLY_1 instead of /SUPPLY_1
- The SIG_NAME property on the pin of the symbol for the component has the value SUPPLY_0 instead of /SUPPLY_0
- The signal you want to be declared as a SUPPLY0 net is named SUPPLY_0 instead of /SUPPLY_0
- The signal you want to be declared as a SUPPLY1 net is named SUPPLY_1 instead of /SUPPLY_1



In the above figure, the signal you want to be declared as a SUPPLY1 net is named SUPPLY_1 instead of /SUPPLY_1. This resulted in the error.

Solution

Ensure that the SUPPLY_1 and SUPPLY_0 signal names are declared as global signals, that is, begin with a forward slash (/) or have a \G suffix.

For more information on declaring SUPPLY0 nets, see SUPPLY_0 on page 627. For more information on declaring SUPPLY1 nets, see SUPPLY_1 on page 628.

ERROR 106: SUPPLY_1 and SUPPLY_0 signals cannot be vectored.

This error has occurred because of the following reasons:

- The SIG_NAME property on the pin of the symbol for the component has a vectored value, say /SUPPLY1<3..0>

- The SIG_NAME property on the pin of the symbol for the component has a vectored value, say `/SUPPLY0<3..0>`
- The signal you want to be declared as a SUPPLY0 net is vectored, say `/SUPPLY0<3..0>`
- The signal you want to be declared as a SUPPLY1 net is vectored, say `/SUPPLY1<3..0>`

In the above figure, the signal you want to be declared as a SUPPLY 1 net is a vectored signal. This resulted in the error.

Solution

Ensure that the /SUPPLY_1 and /SUPPLY_0 signal names are not vectored.

For more information on declaring SUPPLY 0 nets, see SUPPLY_0 on page 627. For more information on declaring SUPPLY 1 nets, see SUPPLY_1 on page 628.

ERROR 110: Range direction needs to be changed to 'TO'.

Description

The range direction specified for the signal, port, or alias is not correct. For example, if the signal name is `DATA<0 DOWNTO 10>`, it is an error because DOWNTO should be used only to indicate a descending range direction.

Solution

Correct the range direction to TO. In the above example, change the signal name to:

`DATA<0 TO 10>`

For more information, see Specifying Ranges for Ports, Signals and Aliases on page 163.

ERROR 111: Range direction needs to be changed to ’DOWNTO’.

Description

The range specified to declare the width of the vectored port, signal or alias is not correct. You have specified a descending range but have given the range direction as ascending.

For example, if you specify the range for a vectored signal as `DATA<7 TO 0>`, this error is displayed because the syntax `TO` in the range specified indicates that it an ascending range, although it is a descending range.

Solution

Use the syntax `DOWNTO` to specify the range direction as descending. Taking the above example, the correct way to specify the descending range is `DATA<7 DOWNTO 0>`.

VHDL_ERROR 112: Value of VHDL_PORT property is set improperly.

Description

The value specified for the VHDL_PORT property is not correct.

Solution

Ensure that the value of the VHDL_PORT property is set to one of the following:

- IN
- AOUT
- BUFFER
- OUT
- INOUT
- LINKAGE

ERROR 113: Port is attached to an unnamed signal.

Description

The port is attached to an unnamed signal.

Solution

You should name the signal that is attached to a port. To do this:

➤ Name the signal with the *Wire* > *Signal Name* command.

ERROR 114: Port is not connected.

Description

The signal to which the port symbol is attached is not connected to any instance on the schematic.

Solution

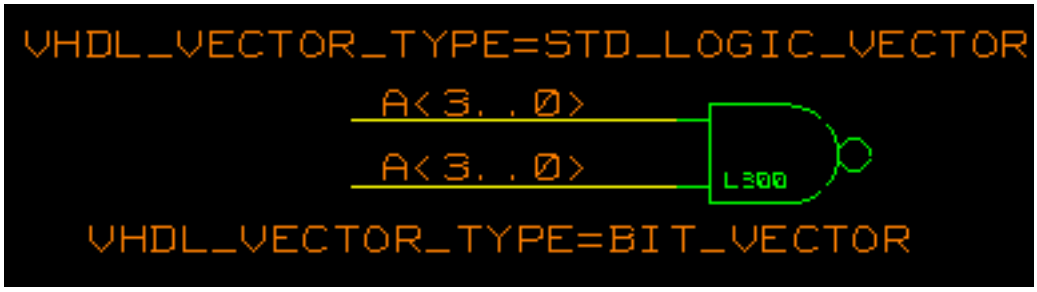
➤ Connect the signal to an instance on the schematic.

For more information on using port symbols in your schematic, see Adding Ports on page 132.

VHDL_ERROR 118: Signal has two different VHDL_VECTOR_TYPE values.

Description

Two different values are specified for the VHDL_VECTOR_TYPE property present on the same signal.



In the above figure, the value assigned to the VHDL_VECTOR_TYPE property on signal A<3..0> attached to the first pin of the LS00 is STD_LOGIC_VECTOR. However, the value assigned to the VHDL_VECTOR_TYPE property on the same signal A<3..0> attached to the second pin of the LS00 is BIT_VECTOR. This resulted in the error.

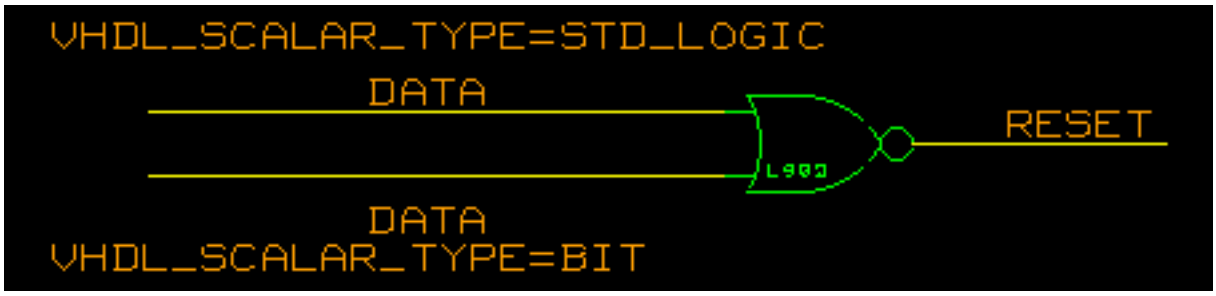
Solution

Ensure that the same value is specified for the VHDL_VECTOR_TYPE property present on the same signal.

VHDL_ERROR 119: Signal has two different VHDL_SCALAR_TYPE values.

Description

Two different values are specified for the VHDL_SCALAR_TYPE property present on the same signal.



In the above figure, the value assigned to the VHDL_SCALAR_TYPE property on signal DATA attached to the first pin of the LS02 is STD_LOGIC. However, the value assigned to the VHDL_SCALAR_TYPE property on the same signal DATA attached to the second pin of the LS02 is BIT. This resulted in the error.

Solution

Ensure that the same value is specified for the VHDL_SCALAR_TYPE property present on the same signals.

ERROR 120: Signal is declared to be both a port and an alias.

Description

The signal connected to an ALIAS symbol is connected to a port.

In the above figure, signal B that is connected to an ALIAS symbol is connected to an OUTPORT symbol, resulting in an error.

Solution

You should not connect a signal that is connected to an `ALIAS` symbol to a port.

For more information on using port and alias symbols in your schematic, see Adding Ports on page 132, and Creating an Alias for a Signal on page 137.

WARNING 121: Port has two different port modes. Port being declared as inout.

Description

Two different port modes have been declared for the port. This resulted in the port being declared as INOUT.

Solution

Check the properties on the pins of the port to ensure that only one port mode is declared for the port. To to this:

1. Choose *Text* > *Attributes*.
2. Click on a pin of the port to display the Attributes dialog box.
3. Verify if the VLOG_MODE or VHDL_MODE property has been used to declare more than one port mode.
4. Perform steps 2 and 3 above for each pin of the port.

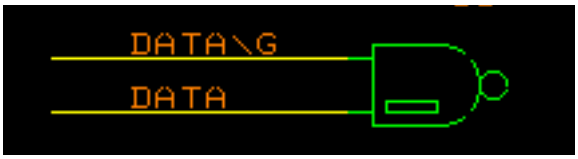
   Note: If the port has several pins, the VLOG_MODE or VHDL_MODE property has to be attached to only one of the pins of the port.

For more information, see Declaring Port Modes on page 183.

WARNING 122: Signal is a global signal in one place but not in the other place. The signal will be treated as a global signal at both the places.

Description

The signal that is declared as a global signal (for example, DATA\G or /DATA is a global signal) in one place in the schematic is not declared as a global signal in another place in the schematic (for example, the signal is declared as DATA and not DATA\G in the other place). The signal will be treated as a global signal at both the places.



In the above figure, signal DATA has the same base name (DATA) as the global signal DATA\G resulting in the error.

Solution

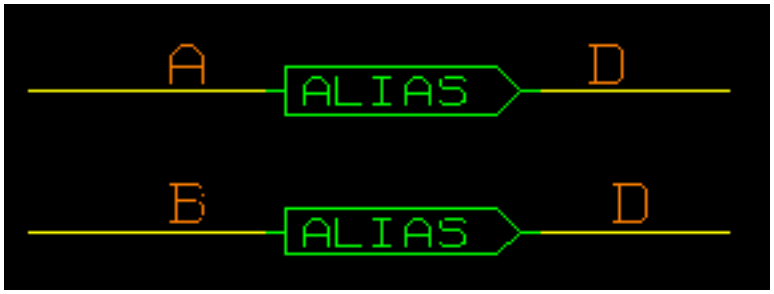
Change the signal names to ensure that no signal has the same name as a global signal.

Note: This error message will not be displayed if a signal connected to an instance that has the HDL_POWER property has the same name as the value of the HDL_POWER property. This is because, Concept HDL treats the signal as a global signal.

ERROR 123: Same alias is made to two different signals.

Description

The ALIAS symbol in the Standard library is used to specify another name for a signal. If the same alias is made to two different signals, this error occurs.



In the above figure, signal A is aliased to signal D and signal B is also aliased to signal D. In this case, signal A is driving signal D and signal B is also driving signal D. This results in the error.

Solution

Ensure that the same alias is not made to two different signals.

For more information on ALIAS symbols, see Creating an Alias for a Signal on page 137.

ERROR 124: Signal is declared to be both a scalar and a vector.

Description

The signal has been declared to be both a scalar and a vectored signal. This error is displayed if, for example, a schematic has two signals `DATA` (a scalar signal) and `DATA <7..0>` (a vectored signal).

Solution

In a schematic you cannot have a signal with the same name declared as a scalar and a vectored signal. Change the signal names in the schematic to ensure that no signal with the same name is not declared as scalar and vector.

ERROR 126: Identifier is used as both a PATH value and a signal name.

Description

The name of the signal is the same as `page<page_number>_<value of PATH property on any instance>`.

When Concept HDL generates the netlist for the design, it writes each component instance in the netlist as: `page<page_number>_<value of PATH property on the instance>`. For example, if the value of the PATH property of an instance on page 1 of the schematic is `i1`, the instance is written in the netlist as `page1_i1`. If the *Check InstanceVs Signal* check box in the *Verilog Netlist* or *VHDL Netlist* dialog box is selected, Concept-HDL displays this error message for every signal that has the same name as `page<page_number>_<value of PATH property on any instance>`.

For more information on setting up the options for netlisting the design, see Chapter 15, “Netlisting Your Design.”

Solution

Change the signal name to ensure that it does not have the same name as `page<page_number>_<value of PATH property on any instance>`.

WARNING 127: Identifier is used as both a component name and a signal name.

Description

The signal name is the same as the name of a component used in the schematic page.

For example, if you have used the `ls04` component in the schematic page and also have a signal named `ls04` in the schematic page, this error occurs.

Solution

Change the signal name to ensure that it is not the same as the name of any component used in the schematic page.

ERROR 128: Net has two port symbols connected to it.

Description

The signal directly connects one port symbol to another port symbol.

If a signal is connected to a port, Concept HDL names the port using the signal name. If a signal directly connects two port symbols, Concept HDL will not be able to determine the type of the port.

In the above figure, signal A directly connects an INPORT symbol to an OUTPORT symbol, resulting in an error.

Solution

Ensure that the signal does not directly connect two port symbols.

For more information on port symbols, see Adding Ports on page 132.

ERROR 129: A global signal cannot also be a port.

Description

The signal connected to the port is a global signal.

In the above figure, signal RESET\G is connected to an INPORT symbol, resulting in an error.

Solution

You must not connect a global signal to a port symbol.

For more information on port symbols, see Adding Ports on page 132. For more information on global signals, see Global Signals on page 140.

ERROR 131: You cannot tap from an unnamed signal.

Description

The signal to which the TAP symbol is attached is unnamed. You should name the signal that is attached to a TAP symbol.

Solution

➤ Name the signal with the *Wire > Signal Name* command.

For more information on using TAP symbols in your schematic, see TAP Symbols on page 616.

ERROR 132: You cannot tap from an unconnected signal.

Description

The signal to which the TAP symbol is attached is not connected to any instance on the schematic.

Solution

➤ Connect the signal to an instance on the schematic.

For more information on using TAP symbols in your schematic, see TAP Symbols on page 616.

ERROR 136: The signal coming out of the concatenation symbol must be unnamed.

Description

The signal coming out of the CONCAT symbol must be unnamed.

Solution

1. Choose *Edit* > *Delete*.
2. Click on the signal name to delete it.

For more information on using CONCAT symbols in your schematic, see CONCAT Symbols on page 621.

ERROR 137: Each pin on a concatenation symbol must be connected to a signal.

Description

One or more pins on a CONCAT symbol are not connected to a signal.

Solution

Ensure that all the pins on the CONCAT symbol are connected to a signal.

For more information on using CONCAT symbols in your schematic, see CONCAT Symbols on page 621.

ERROR 144: Alias symbol has an unconnected pin.

Description

One of the pins of the ALIAS symbol is not connected to any signal.

Solution

Ensure that both the pins of the ALIAS symbol are connected to signals.

For more information on ALIAS symbols, see Creating an Alias for a Signal on page 137.

ERROR 145: Pin on alias symbol has an unnamed signal attached.

Description

An unnamed signal is attached to the pin of the ALIAS symbol. You should name the signal that is attached to the pin.

Solution

➤ Name the signal with the *Wire > Signal Name* command.

For more information on using the ALIAS symbol in your schematic, see Creating an Alias for a Signal on page 137.

ERROR 146: Signal coming out of an alias symbol is also a port.

Description

The signal coming out of the ALIAS symbol is connected to a port symbol.

In the above figure, signal B is connected to an OUTPORT signal, resulting in an error.

Solution

You should not connect the signal coming out of an ALIAS symbol directly to a port symbol.

For more information on ALIAS symbols, see Creating an Alias for a Signal on page 137.

ERROR 147: Signal coming out of an alias symbol cannot be a global signal.

Description

The signal coming out of the ALIAS symbol is a global signal.

Solution

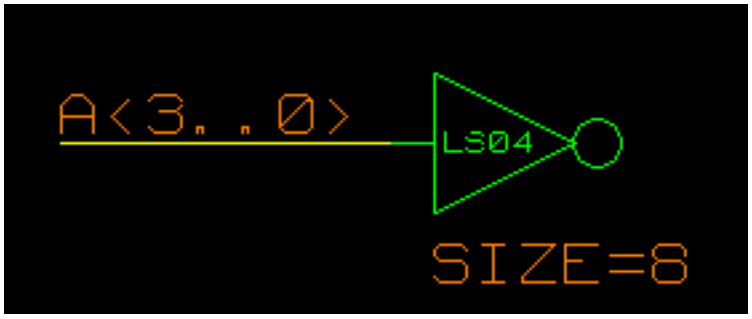
Ensure that the signal coming out of the ALIAS symbol is not a global signal.

For more information on ALIAS symbols, see Creating an Alias for a Signal on page 137. For more information on global signals, see Global Signals on page 140.

ERROR 150: Signal connected to pin has incorrect width.

Description

The value of the SIZE property specified on the component determines the width of the pins on the component. If the width of the signal and the width of the pin of the component to which the signal is connected are not the same, this error occurs.



In the above figure, the value of the SIZE property on the component is 8. However, since the width of signal A is 4, this error occurs.

Solution

Ensure that the value of the SIZE property specified on the component is the same as the width of the signal.

ERROR 151: Entity declaration for instance declares a port that is not on the instance.

Description

The entity declaration for the instance (`/entity/verilog.v` or `/entity/vhdl.vhd` file) declares a port that is not present on the instance.

Solution

Open the symbol for the component in Concept HDL or Part Developer and add the port on the symbol, if required. Then save the symbol. This will bring your symbol and entity ports in sync.

ERROR 152: Port on instance does not exist in entity declaration for instance.

Description

The port present on the instance does not exist in the entity declaration for the instance (`/entity/verilog.v` or `/entity/vhdl.vhd` file).

Solution

Open the symbol for the component in Concept HDL or Part Developer and delete the port on the symbol, if it is not required. Then save the symbol. This will bring your symbol and entity ports in sync.

ERROR 153: Port on instance is vectored but port in entity declaration for instance is not.

Description

The port on the instance is a vectored port but the same port in the entity declaration for the instance (`/entity/verilog.v` or `/entity/vhdl.vhd` file) is not a vectored port.

Solution

Open the symbol for the component in Concept HDL or Part Developer and correct the port on the symbol of the instance. Then save the symbol. This will bring your symbol and entity ports in sync.

ERROR 154: Port on instance is scalar but port in entity declaration for instance is not.

Description

The port declared on the symbol instance is a scalar port but the same port in the entity declaration for the instance (`/entity/verilog.v` or `/entity/vhdl.vhd`) is not scalar.

Solution

Open the symbol for the component in Concept HDL or Part Developer and correct the port on the symbol of the instance. Then save the symbol. This will bring your symbol and entity ports in sync.

ERROR 155: Range direction for port on instance conflicts with port in entity declaration for instance.

Description

The range direction for the port on the instance conflicts with the range direction for the port in the entity declaration for the instance (the `verilog.v` or `vhdl.vhd` file in the `entity` view of the instance).

For example, if the range on the port on the instance is `<0 to 3>` and the range for the port in the entity declaration for the instance is `<3 downto 0>`, this error occurs.

Solution

Open the symbol for the component in Concept HDL or Part Developer and save the symbol. This will bring your symbol and entity ports in sync.

ERROR 156: Instance port and entity port modes are incompatible.

Description

There is a mismatch between the port mode defined in the component declaration for the instance and the entity declaration associated with the component.

The following port modes can be declared using the VHDL_MODE property:

- IN
- OUT
- BUF
- INOUT

Solution

Verify that the following mismatches do not exist in the port mode declared in the instance and the entity:

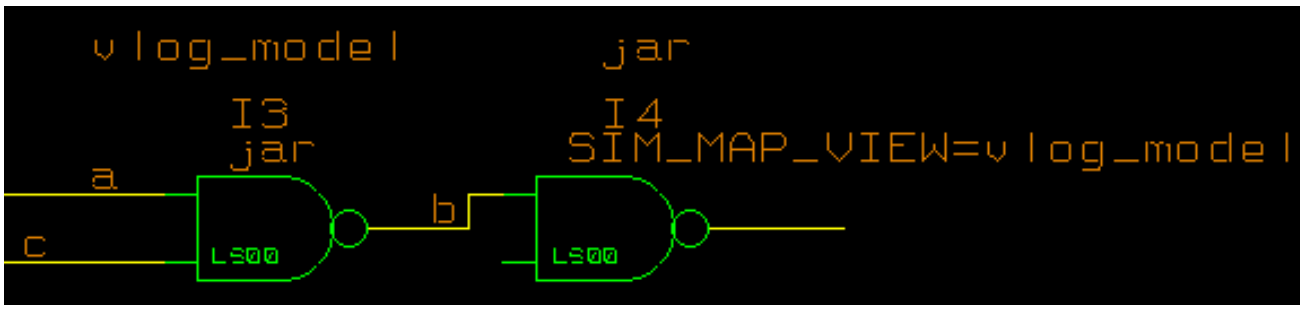
- PORT declared as IN in entity and OUT, INOUT, or BUF on instance
- PORT declared as OUT in entity and IN, INOUT, or BUF on instance
- PORT declared as BUF in entity and IN, OUT, or INOUT on instance
- PORT declared as INOUT in entity and BUF on instance

For more information, see Declaring Port Modes on page 183.

ERROR 158: Sizeable pin cannot be represented in Verilog because it is partly unconnected.

Description

This error is generated when you use wrappers to simulate a sizeable part that has unconnected pins, and has either `SPLIT_INST_NAME` or `SPLIT_INST` property attached to it. The error is generated because Concept HDL netlister is unable to determine the width of open pins.



Solution

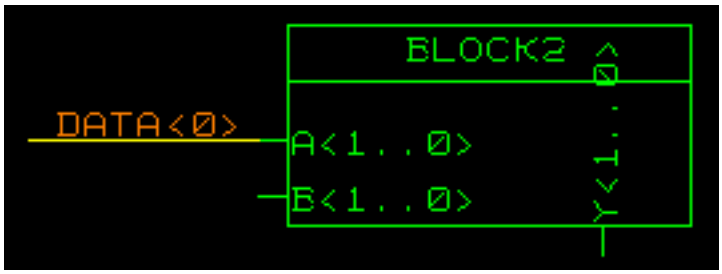
To remove the error, perform one of the following steps:

- Do not have unconnected pins in the schematic.
- Use `SPLIT_INST` and `SPLIT_INST_NAME` properties only on asymmetrical parts.

ERROR 164: Pin width is greater than attached signal width.

Description

The width of the signal attached to the pin is lesser than the width of the pin.



In the above figure, the width of pin `A<1..0>` is greater than the width of the signal `DATA<0>` attached to it, resulting in an error.

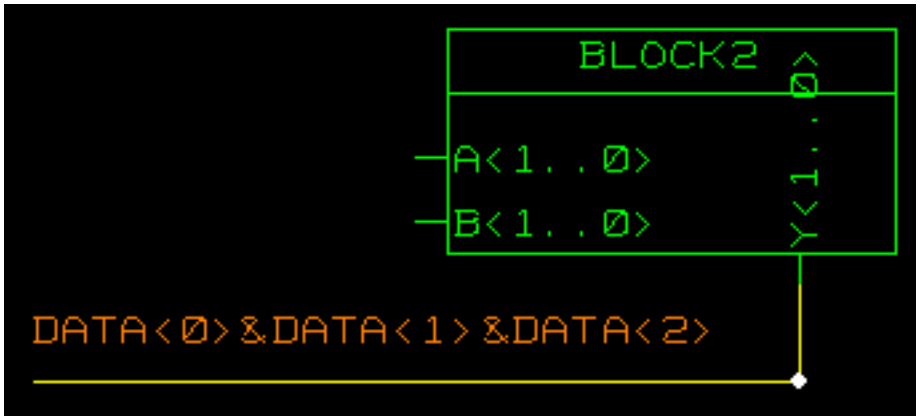
Solution

Ensure that the width of the signal attached to the pin is the same or an integral multiple of the width of the pin.

ERROR 165: Concatenated signal width must match pin width.

Description

The width of the concatenated signal attached to the pin is not the same as the width of the pin.



In the above figure, the width of pin Y<1..0> is lesser than the width of the concatenated signal DATA<0>&DATA<1>&DATA<2> attached to it, resulting in an error.

Solution

Ensure that the width of the concatenated signal attached to the pin is the same as the width of the pin.

In the above example, the width of the concatenated signal should be changed to DATA<0>&DATA<1>.

For more information on concatenating signals, see Signal Concatenations on page 147.

ERROR 166: Attached signal width is not an integer multiple of pin width.

Description

The width of the vectored signal is not an integer multiple of the width of a sizable pin to which it is attached.

For example, suppose that a vectored signal S is attached to pin A of component LS00. The property SIZE=4 is attached to pin A. In this case, the pin width is 1, but if the signal width is not 4 (not an integer multiple of the pin width 4 x 1), this error occurs.

Solution

Change the width of the signal to be an integer multiple of the width of the sizable pin.

ENTITY ERROR 169: Port mode for the pin on the symbol is different from that on the pin of the instance. Modify the port mode to make it the same."

Description

The port mode for the pin on the symbol and the corresponding pin on the instance are different.

Suppose that you have created a symbol from a schematic or a schematic from a symbol. This error message will appear if, later on, you do one of the following:

- Change the port mode for a pin on the symbol but do not make the same change in the port mode of the corresponding pin on the instance.

- Change the port mode for a pin on the instance but do not make the same change in the port mode of the corresponding pin on the symbol.

For example, if the port mode specified on the pin on the symbol is `VHDL_MODE=IN` and the port mode for the corresponding pin on the instance is `VHDL_MODE=OUT`, this error occurs.

Solution

The port mode for a pin on the symbol must always be the same as the the port mode of the corresponding pin on the instance.

- If the pin on the symbol has the correct port mode, do one of the following:
  - Choose *Tools > Generate View* to regenerate the schematic from the symbol.
  - Open the schematic in Concept HDL, correct the port mode on the pin on the instance to make it the same as the port mode of the corresponding pin on the symbol and save the schematic.
- If the pin on the instance has the correct port mode, do one of the following:
  - Choose *Tools > Generate View* to regenerate the symbol from the schematic.
  - Open the symbol in Concept HDL, correct port mode on the pin on the symbol to make it the same as the port mode of the corresponding pin on the instance and save the symbol.

The allowed values for the `VHDL_MODE` property on a pin are `IN`, `OUT`, `INOUT`, and `BUFFER`. The allowed values for the `VLOG_MODE` property on a pin are `INPUT`, `OUTPUT`, and `INOUT`. For more information, see Declaring Port Modes on page 183.

If you want to specify the port mode for an interface signal on a schematic, attach one of the symbols `INPORT`, `OUTPORT`, `IOPORT` or `BUFPORT` from the Standard library to the interface signal.

ERROR 174: Output of tap is unconnected.

Description

The output pin of the tap symbol (CTAP, BIT TAP, LSB TAP or MSB TAP symbols) is not connected to any signal.

In the above figure, the output pin of the CTAP symbol is not connected to any signal, resulting in an error.

Solution

Connect the output pin of the tap symbol to a signal.

For more information on tap symbols, see TAP Symbols on page 616.

WARNING 177: Entity declaration for part does not exist in library.

Description

When Concept HDL generates the VHDL netlist for a schematic Concept HDL cross checks the entity and symbol views. it also generates a component declaration for each component used in the schematic. To generate this component declaration, Concept HDL reads the entity declaration associated with the component. For example, if the part `NAND2` from the `/usr/libs/lsttl` library is instantiated in a schematic, Concept HDL reads the `/usr/libs/lsttl/nand2/entity/vhdl.vhd` file to generate the VHDL component declaration.

If the entity declaration file is not found this warning message is displayed.

The VHDL netlist still gets created and the component declaration section in the VHDL netlist (`/sch_1/vhdl.vhd`) is constructed from the symbol view of the component (`/sym_1/symbol.css`). Since the symbol view does not contain information about the port mode of the component, Concept-HDL declares all the ports as `inout` in the component declaration section of the netlist.

Solution

Open the symbol for the component in Concept HDL or Part Developer and save it. The entity declaration file (`/entity/vhdl.vhd`) for the component are created in the entity view of the component.

ERROR 178: Port exists in the entity but not on the instantiated symbol. Please rewrite the necessary pages.

Description

The entity declaration for the instance (`/entity/verilog.v` or `/entity/vhdl.vhd` file) declares a port that is not present on the symbol for the instance.

Solution

Open the symbol for the component in Concept HDL or Part Developer and add the port on the symbol, if required. Then save the symbol. This will bring your symbol and entity ports in sync.

To save the symbol in Concept HDL, do the following:

1. In Concept HDL, choose *File > Open*.

   The *View Open* dialog box appears.

2. Select the library in which the component exists in the *Library* drop-down.

   The list of components in the library are displayed.

3. Select the component for which the error occurred.

   The component name is displayed in the *Cell* field.

4. Select *Symbol* from the *View* drop-down.

5. Click *Open* to view the symbol in Concept HDL.

6. Choose *File > Save*.

ERROR 179: Two signal names are attached to this net.

Description

Two signal names are attached to the wire.

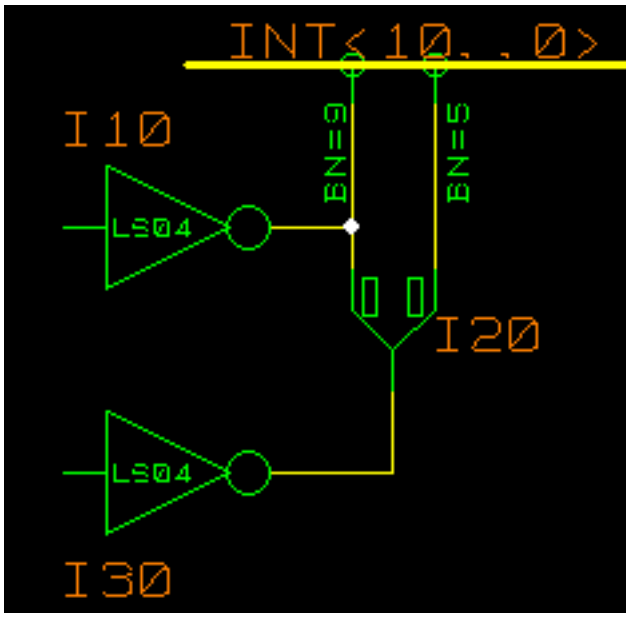
Solution

Remove one of the signal names attached to the wire.

ERROR 181: Signals on both the sides of the MERGE/TIE symbol are connected to driver pins of other instances.

Description

The signals on both the sides of the MERGE or TIE symbol are connected to driver (output) pins of other instances.



In the above figure, the signal connected to the input pin of the 2 MERGE symbol is connected to the driver (output) pin of instance `I10` of the `ls04` component. The signal connected to the output pin of the MERGE symbol is also connected to the driver pin of instance `I30` of the `ls04` component. This error occurs because the signals on both the sides of the MERGE symbol are connected to driver pins of instances of the `lso4` component.

Solution

If the signals on one side of the MERGE or TIE symbol are connected to driver pins of other instances, ensure that the signals on the other side of the MERGE or TIE symbol are not connected to driver pins of other instances.

ERROR 182: Signal on one side of the MERGE/TIE symbol is a global signal and signal on the other side is connected to driver pin of another instance.

Description

The signal on one side of the MERGEor TIE symbol is a global signal and the signal on the other side of the MERGEor TIE symbol is connected to the driver pin of another instance.

Solution

If the signal on one side of the MERGEor TIE symbol is a global signal, ensure that the signal on the other side of the MERGEor TIE symbol is not connected to the driver pin of other instances.

ERROR 183: Both sides of the MERGE/TIE symbol are connected to global signals.

Description

Both the sides of the MERGE or TIE symbol is connected to global signals.

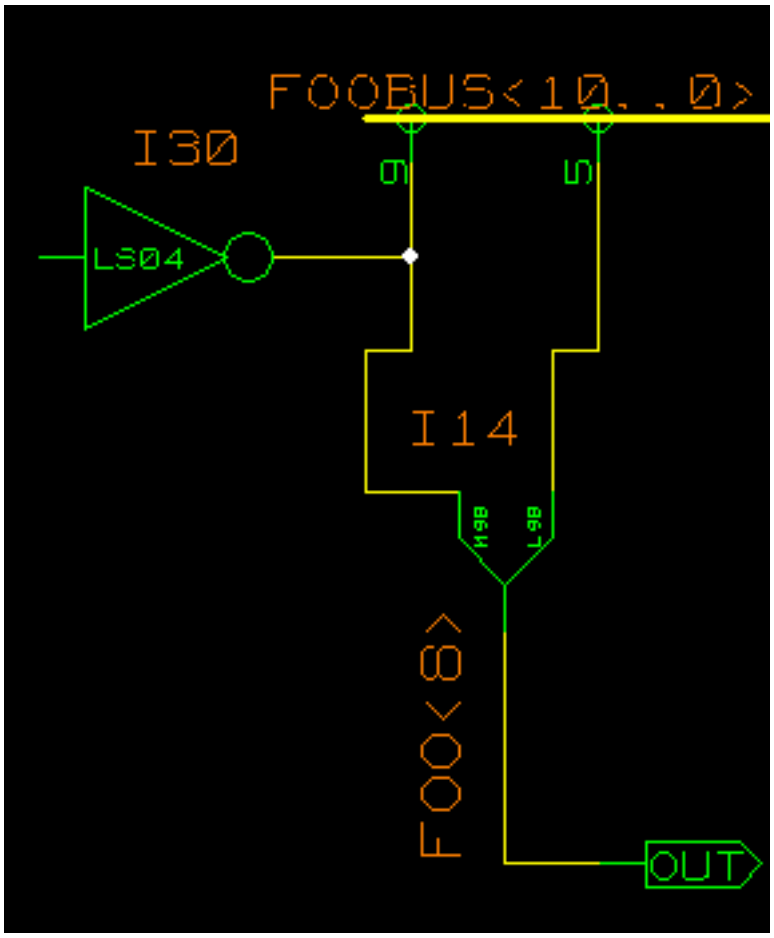
Solution

Ensure that both the sides of the MERGE or TIE symbol are not connected to global signals.

WARNING 184: A OUTPORT symbol in the schematic must not be connected to MERGE/TIE symbol.

Description

This error occurs because the output of the MERGE or TIE symbol is directly connected to an OUTPORT symbol.



In the above figure, the output of the 2 MERGE symbol is connected to an OUTPORT symbol. This results in an error.

Solution

Do not connect the output of a MERGE or TIE symbol directly to an OUTPORT symbol.

ERROR 185: One of the pins of MERGE/TIE symbol is unconnected.

Description

One of the pins of the MERGE or TIE symbol is not connected to any signal.

Solution

Connect a signal to the pin of the MERGE or TIE symbol.

ERROR 187: Signals attached to MERGE/TIE symbol have parameterized width.

Description

The signal attached to the MERGE or TIE symbol has parametrized width. For example, if a parametrized signal `CLOCK<SIZE-1..0>` is attached to the MERGE or TIE symbol, this error occurs.

Solution

Ensure that the signals connected to the MERGE or TIE symbol do not have parametrized width.

ERROR 188: Signals attached to each side of MERGE/TIE symbol have different width.

Description

The width of the signals attached to each side of the MERGE or TIE symbol are not the same. For example, if the width of the signal attached to one side of the TIE symbol is `<3..0>` and the width of the signal attached to the other side of the TIE symbol is `<7..0>`, this error will occur.

The sum of the size of the signals connected to the input pins of a MERGE symbol must be equal to the size of the signal connected to the output pin of the MERGE symbol.

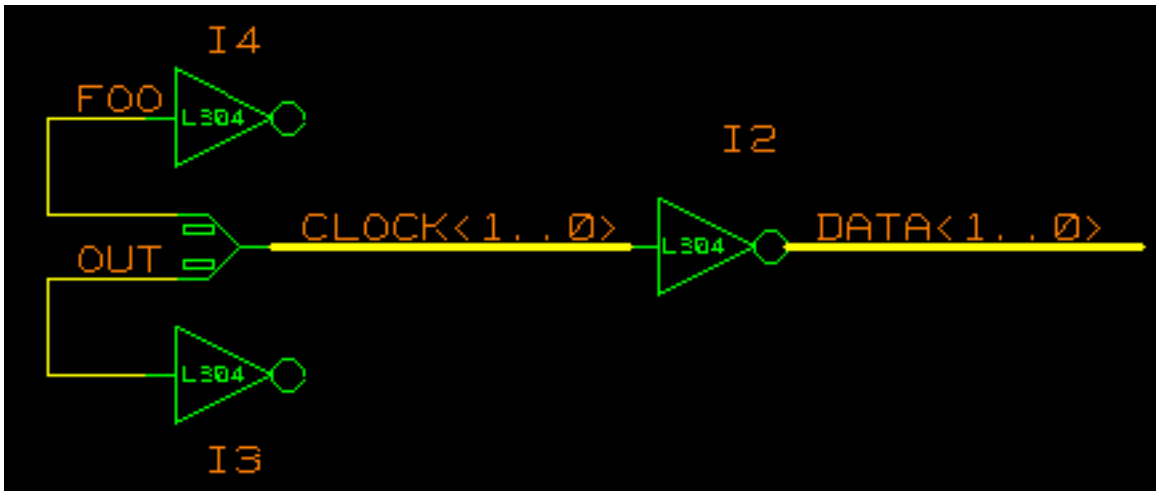
Solution

Ensure that the width of the signals attached to each side of the MERGE or TIE symbol are the same.

ERROR 190: Signals on both the sides of the MERGE/TIE symbol are undriven.

Description

The signals on both the sides of the MERGE/TIE symbol are undriven.



In the above figure, the signal `FOO` connected to the input pin of the 2 MERGE symbol is connected to the receiver (input) pin of instance `I4` of the `ls04` component. The signal `CLOCK<1..0>` connected to the output pin of the MERGE symbol is also connected to the receiver (input) pin of instance `I2` of the `ls04` component. This error occurs because the signals on both the sides of the MERGE/TIE symbol are undriven.

Solution

Ensure that the signals on one side of the MERGE or TIE symbol are connected to driver (output) pins of other instances and that the signals on the other side of the MERGE or TIE symbol are not connected to driver pins of other instances. In other words, if the signals on one side of the MERGE or TIE symbol are connected to driver pins of other instances, ensure that the signals on the other side of the MERGE or TIE symbol connected only to the receiver (input) pins of other instances.

WARNING 191: Cannot place pin properties on a pin with parameterized width if other pins on the instance have the same basename.

Description

If a property is attached to a vectored pin of an instance that has parametrized width and another pin of the instance has the same base name, this warning is displayed.

For example, suppose there are two pins A<1..0> and A<2> on a component. If a property is attached to pin A<1..0>, it is not clear if you want the same property on pin A<2> also. This results in the warning message being displayed.

Solution

Ensure that pins on a component do not have the same base name. Taking the above example, rename pin A<2> as B<2>.

WARNING 192: Cannot place properties on specific bits of a signal which has parameterized width.

Description

If a property is attached to a bit of a vectored signal that has parametrized width and another signal on the design has the same base name, this warning is displayed.

For example, suppose there are two signals A<3..0> and A<2> in a design. If a property is attached to signal A<2>, it is not clear if you want the same property on signal A<3..0> also. This results in the warning message being displayed.

Solution

Ensure that signals in your design do not have the same base name. Taking the above example, rename pin A<2> as B<2>.

ERROR 197: Property on declarations symbol has incorrect value.

Description

A property defined on the DECLARATIONS symbol has an incorrect value.

Solution

See the details of the error message in the *Markers* window to know the name of the property that has the incorrect value. Refer to the *PCB Systems Properties Reference* for information on the supported values for the property.

ERROR 198: Signal syntax is incorrect.

Description

There is a syntax error in the signal name. For example, if the signal name is ~DATA, this error is displayed.

Solution

Ensure that the signal name complies with the conventions for naming of signals in Concept-HDL. For more information, see Signal Naming Conventions on page 126.

WARNING 211: Size Property not present on instance. Assuming a value of 1.

Description

This error is generated when an instance has either HDL_REPLICATE = TRUE, HDL_LSBTAP = TRUE, or HDL_MSBTAP = TRUE property attached to it without the SIZE property.

The HDL_REPLICATE property, which is attached to the origin of a symbol, classifies the symbol as REPLICATE. The REPLICATE symbol is used while making models for sizeable parts. Therefore, the HDL_REPLICATE property must always be used along with the SIZE property.

Similarly, with HDL_LSBTAP and HDL_MSBTAP properties, the SIZE property is required to specify the width of the tapped signal.

Solution

If the SIZE property is not specified, by default the tool assumes the value to be 1. To change the size to a value other than 1:

1. Select *Text > Attributes*.

2. Click on the instance to which the SIZE property is to be added.

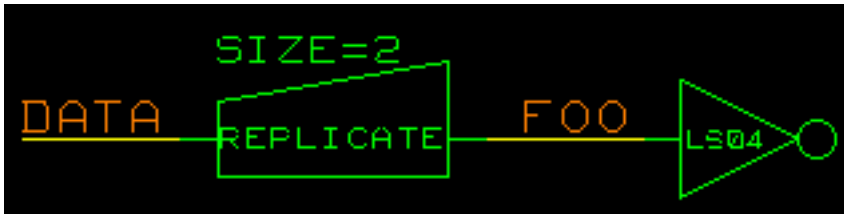
   The *Attributes* dialog box appears.

3. Click *Add*.
4. Enter the *Property Name* as SIZE and the *Value* as 2.
5. Click OK.

ERROR 212: Net connected to the output of Replicate instance should be unnamed or have the same width as the Replicate instance.

Description

The net connected to the output of the REPLICATE symbol is named or does not have the same width as the REPLICATE symbol.



In the above figure, the REPLICATE symbol has the width 2 (denoted by the SIZE=2 property). This error occurs because the signal FOO connected to the output pin of the REPLICATE symbol does not have the same width as the REPLICATE symbol.

Solution

Do one of the following:

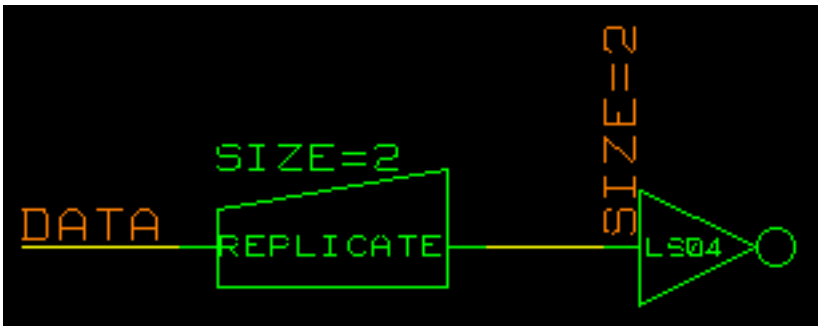
- Ensure that the signal connected to the output pin of the REPLICATE symbol has the same width as the REPLICATE symbol.

  In the above example, change the name of the signal to FOO<1..0>.

- Do not name the signal connected to the output pin of the REPLICATE symbol.

  If you do not name the signal connected to the output pin, ensure that the pin of the instance to which the output signal is connected has the same width as the REPLICATE

symbol. In the above example, if the output signal is not named, add the `SIZE=2` property on the input pin of the `ls04` component.



WARNING 217: Bit property not present on instance. Assuming a value of 0

Description

If the value of the HDL_BITTAP property specified on an instance is TRUE, Concept HDL reads the value of the BIT property specified on the instance to determine the bit to be tapped from the bus connected to the instance. If the BIT property is not present on the instance, Concept HDL assumes that 0 bits have to be tapped from the bus and displays this warning message.

Solution

Add the BIT property on the instance to specify the bit to be tapped from the bus connected to the instance.

ERROR 222: Error in symbol files.

Description

An error occurred when reading the `symbol.css` file in the symbol view of the component.

Solution

Open the symbol in Concept HDL and save it to regenerate the symbol files. You can also open the symbol in the Part Developer tool and save it to regenerate the symbol files. For more information, see the *Part Developer User Guide*.

To save the symbol in Concept HDL, do the following:

1. In Concept HDL, choose *File > Open*.

   The *View Open* dialog box appears.
2. Select the library in which the component exists in the *Library* drop-down.

   The list of components in the library are displayed.
3. Select the component for which the error occurred.

   The component name is displayed in the *Cell* field.
4. Select *Symbol* from the *View* drop-down.
5. Click *Open* to view the symbol in Concept HDL.
6. Choose *File > Save*.

ERROR 230: Net widths on both sides of the merge body do not match.

Description

The combined width of the signals connected to the input pins of the MERGE symbol is not the same as the width of signal connected to the output pin of the MERGE symbol.

Solution

Correct the width of the signal connected to the output pin of the MERGE symbol.

For more information on using MERGE symbols in your schematic, see Merge Symbols on page 150.

ERROR 231: Symbol pin is wider than the entity port.

Description

The width of the pin on the symbol is wider than the width of the port in the entity. For example, if the width of the pin on the symbol is `a<3..0>` and the width of the port specified in the `verilog.v` or `vhdl.vhd` file in the entity view of the part is `a<2..0>`, this error occurs.

Solution

Open the symbol in Concept HDL and correct the width of the pin on the symbol. You can also open the symbol in the Part Developer tool and correct the width of the pin on the symbol. For more information, see the *Part Developer User Guid*e.

To correct the width of the pin on the symbol in Concept HDL, do the following:

1. In Concept HDL, choose *File > Open*.

   The *View Open* dialog box appears.
2. Select the library in which the component exists in the *Library* drop-down.

   The list of components in the library are displayed.
3. Select the component for which the error occurred.

   The component name is displayed in the *Cell* field.
4. Select *Symbol* from the *View* drop-down.
5. Click *Open* to view the symbol in Concept HDL.
6. Correct the width of the pin of the symbol.
7. Choose *File > Save*.

ERROR 234: Different component uses same SPLIT_INST_NAME/SPLIT_INST prop value. Use different prop value for different components.

Description

The SPLIT_INST_NAME property value for two different components are the same.

Solution

The SPLIT_INST_NAME property is used on all split components of a large pin count device that have to be merged into a single instance in the netlist. The value of the SPLIT_INST_NAME property must be the same on such split components. This error occurs if the value of the SPLIT_INST_NAME property on split components of two different large pin count devices is the same.

For example, suppose there are two large pin count devices `ASYM_PART` and `ASYM_PART1`. The device `ASYM_PART` is split into two components-`INST1` and `INST2`. The device `ASYM_PART1` is split into three components-`INST_A`, `INST_B`, and `INST_C`. If the same

value is specified for the SPLIT_INST_NAME property on component INST1 of device ASYM_PART and component INST_A of device ASYM_PART1, this error occurs.

ERROR 260: Two assertion character - and * used in the signal name, it is not allowed. Use only one assertion character

Description

You can use the * character as a suffix or the - character as a prefix in a signal or a pin name to declare a low-asserted signal or pin. If both the assertion characters (- and *) are used in the signal or pin name, this error occurs.

Solution

Use only one of the assertion characters in the signal or pin name.

Note: You can also use a _N suffix in a signal or a pin name to declare a low-asserted signal or pin. Cadence recommends that you use a _N suffix to indicate a low-asserted signal or pin.

ERROR 264: Property (SIZE/HAS_FIXED_SIZE/TIMES) can have only integer value. Ignoring this value and using 1 as the default value.

Description

The SIZE, HAS_FIXED_SIZE, and TIMES properties can have only integer values. Although you can assign alphanumeric values to these properties, only those alphanumeric values where the first letter is a numeral are allowed. In such cases, only the numeric part of the value is considered and rest is ignored.

Property Value	Concept HDL netlister.....
SIZE = 3A	Sets the size to 3.
Size = 3A3	Sets the size to 3.
SIZE = A3	Generates an error.

Solution

Assign numeric values to SIZE, HAS_FIXED_SIZE, and TIMES properties.

ENTITY_ERROR 267: Port range specified in the schematic and symbol is different. Modify schematic/symbol to make port range same.

Description

The range of a port on the symbol is different from the range of the corresponding port on the schematic.

Suppose that you have created a symbol from a schematic or a schematic from a symbol. This error message will appear if, later on, you do one of the following:

- Change the range of a port on the symbol but do not make the same change in the range of the corresponding port on the schematic.
- Change the range of a port on the schematic but do not make the same change in the range of the corresponding port on the symbol.

For example, if the port range specified on the symbol is `ADDRESS<3..0>` and the port range on the schematic is `ADDRESS<7..0>`, this error occurs.

Solution

The range of a port on the symbol must always be the same as the range of the corresponding port on the schematic.

- If the symbol has the correct port range, do one of the following :
  - Choose *Tools > Generate View* to regenerate the schematic from the symbol.
  - Open the schematic in Concept HDL, correct the port range on the schematic to make it the same as the port range on the symbol and save the schematic.
- If the schematic has the correct port range, do one of the following :
  - Choose *Tools > Generate View* to regenerate the symbol from the schematic.
  - Open the symbol in Concept HDL, correct the port range on the symbol to make it the same as the port range on the schematic and save the symbol.

ENTITY_ERROR 268: Port is specified vectored in the schematic but scalar on symbol. Modify schematic/symbol to make port consistent.

The port on the schematic is a vectored port, but the corresponding port on the symbol is a scalar port.

Suppose that you have created a symbol from a schematic or a schematic from a symbol. This error message will appear if, later on, you do one of the following:

- Change a vectored port to a scalar port on the symbol, but do not make the same change for the corresponding port on the schematic.
- Change a scalar port to a vectored port on the schematic, but do not make the same change for the corresponding port on the symbol.

For example, if the port on the schematic is ADDRESS<7..0> (a vectored port) and the port on the symbol is ADDRESS (a scalar port), this error occurs.

Solution

The ports on the schematic and the symbol must always be in sync.

- If you want to declare the port as scalar, do one of the following :
  - Choose *Tools > Generate View* to regenerate the schematic from the symbol.
  - Open the schematic in Concept HDL, change the vectored port on the schematic to make it a scalar port and save the schematic.
- If you want to declare the port as vectored, do one of the following :
  - Choose *Tools > Generate View* to regenerate the symbol from the schematic.
  - Open the symbol in Concept HDL, change the scalar port on the symbol to make it a vectored port and save the symbol.

ENTITY_ERROR 269: Port is specified scalar in the schematic but vectored on symbol. Modify schematic/symbol to make port consistent.

Description

The port on the schematic is a scalar port, but the port on the symbol is a vectored port.

Suppose that you have created a symbol from a schematic or a schematic from a symbol. This error message will appear if, later on, you do one of the following:

- Change a vectored port to a scalar port on the schematic, but do not make the same change for the corresponding port on the symbol.
- Change a scalar port to a vectored port on the symbol, but do not make the same change for the corresponding port on the schematic.

For example, if the port on the schematic is ADDRESS (a scalar port) and the port on the symbol is ADDRESS<7..0> (a vectored port), this error occurs.

Solution

The ports on the schematic and the symbol must always be in sync.

- If you want to declare the port as scalar, do one of the following :
  - Choose *Tools > Generate View* to regenerate the symbol from the schematic.
  - Open the symbol in Concept HDL, change the vectored port on the symbol to make it a scalar port and save the symbol.
- If you want to declare the port as vectored, do one of the following :
  - Choose *Tools > Generate View* to regenerate the schematic from the symbol.
  - Open the schematic in Concept HDL, change the scalar port on the schematic to make it a vectored port and save the schematic.

ERROR 275: Two global signals are shorted.

Description

The global signal is shorted with another global signal.

Solution

Two global signals should not be shorted. If you want to short the global signals, add them in the *Allowed Global Shorts* list of the *Concept HDL Options* dialog box. Concept HDL will not display this error message if the global signals listed in the *Allowed Global Shorts* list are shorted. For more information, see Shorting of Global Signals on page 141.

This error message will also be displayed when you netlist the design for digital simulation, if you have specified two shorted global signals CLOCK\G and SWITCH\G in the *Allowed Global Shorts* list and have also added CLOCK\G in the *Supply 0* list and SWITCH\G in the *Supply 1* list in the *Verilog* netlisting options dialog box.

For more information on setting up Verilog netlisting options and netlisting the design for digital simulation, see Netlisting for Digital Simulation on page 376.

WARNING 401: Binding Instance

The view specified by the SIM_MAP_VIEW property for instance <instance_name>, pathval: <path_value> does not exist or doesn't have a valid map file. The specified property, <property_name and value> is being ignored.

Description

This is a warning message that is generated in the following two scenarios:

- There is no view corresponding to the view name specified by the value assigned to the SIM_MAP_VIEW property.
- The view exists but does not have a valid verilog.map file.

In both the above mentioned cases the default configuration is used for binding the component instances.

Solution

To remove this error message you must specify a view name that has a valid map file, or remove the SIM_MAP_VIEW property. To avoid incorrect simulation results, it is recommended that these warnings must be removed before simulating the design.

ERROR 422: Chips File Packaging Error

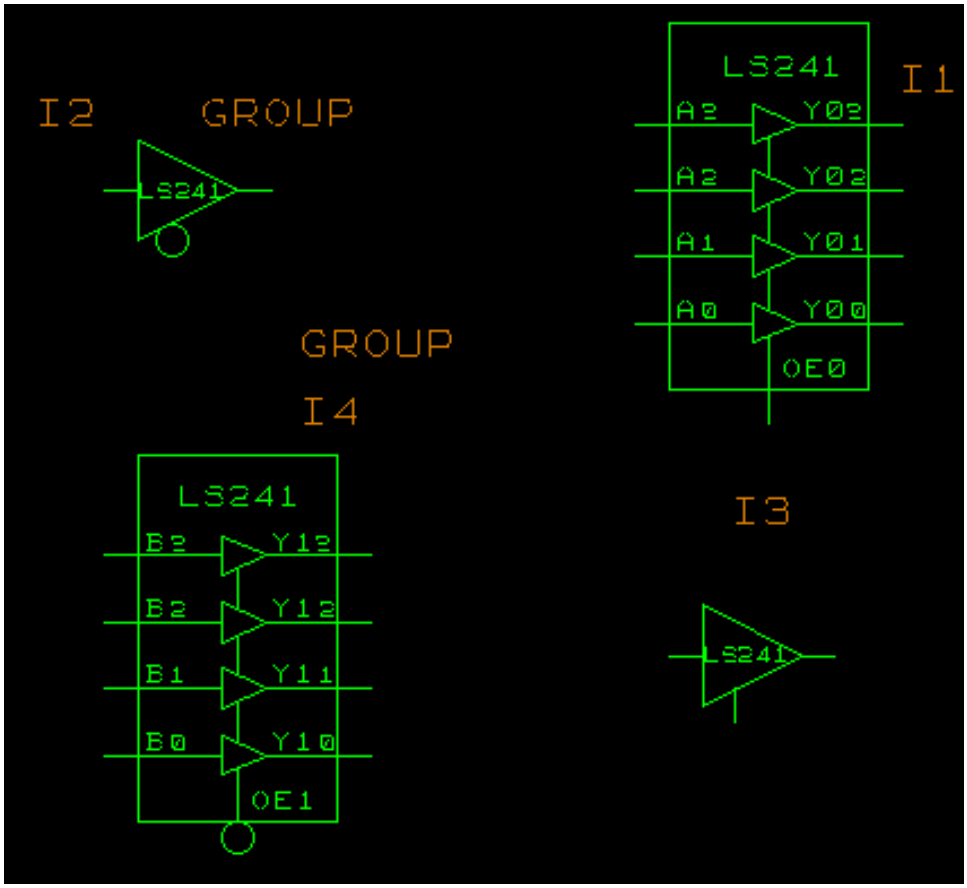
Cannot package primitive instance: <instance_name>, pathval: <path_value>. Ignoring split inst property.

Description

This error is generated when SPLIT_INST or SPLIT_INST_NAME properties are used in a manner that violates the package description in the chips.prt file.

For example, according to the chips.prt file of LS241, from among the parts shown below, I2 and I4 can only be grouped with either I1 or I3. If I2 and I4 are grouped together, Chips File

Packaging Error is generated because you have more than one entry for a pin name which violates the description in the `chips.prt` file.



Solution

Before grouping parts in the same split inst group, read the `chips.prt` file to find out the parts that can be grouped together and then use the `SPLIT_INST` or `SPLIT_INST_NAME` property to group two or more parts in the same split inst group.

ERROR 521: In Specifying Property On Instance

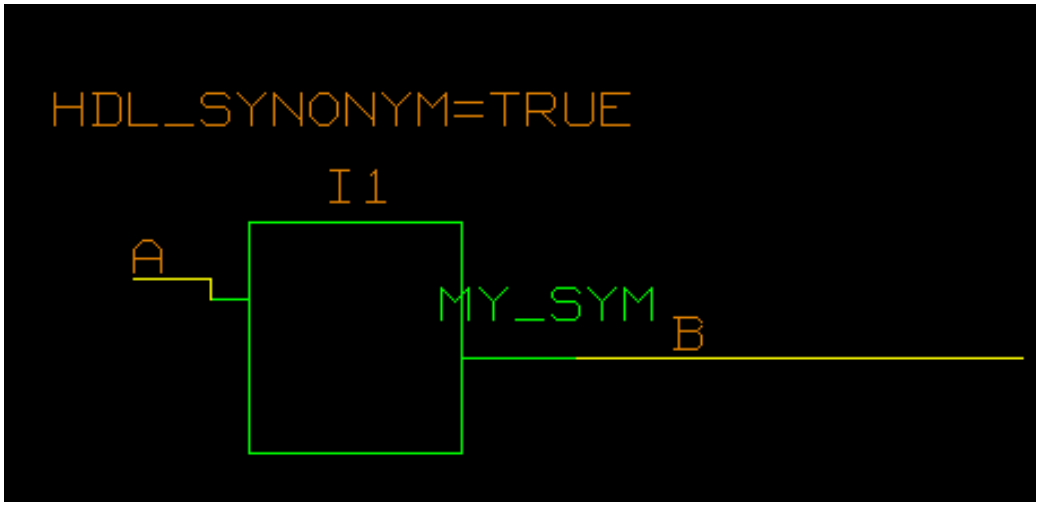
This message appears when ever the properties specified on an instance are not correct. This message may appear either as an error or a warning. Depending on the situation in which the message appears, the message has different explanations. Some of the explanations are listed below:

Incorrect SYNONYM property on instance <component_name>. The SYNONYM property can be specified only on components with exactly 2 pins. The HDL_SYNONYM property is being ignored.

Description

This warning is generated when the HDL_SYNONYM property is attached to a symbol that has more than one input or output pins. This is because the symbols or the parts with HDL_SYNONYM = TRUE are used to specify a different name for the same signal.

A symbol with correct usage of HDL_SYNONYM = TRUE and the corresponding Verilog netlist are shown below:



Verilog Netlist:

```
wire a;
wire  b;

  wire  page1_a;
  wire  page1_b;

  assign page1_a = a;
  assign page1_b = b;

  assign a  = b;
```

Solution

Add HDL_SYNONYM property only to components and symbols that have exactly one input and one output pin. To know more about SYNONYM, see *Using the Standard Library Symbols* in *Concept HDL User Guide*.

Incorrect property specified for instance <instance_name>, pathval <path_value>. The specified property REMOVE = LINK is being ignored. The REMOVE property can only be used with components having exactly one input and one output pin.

Description

This error is generated when REMOVE = LINK is attached to a component that does not satisfy the criterion of one input and one output pin. The REMOVE property is attached only to the two pin components. Resistor packs with one input and multiple outputs are the only exception to this rule. You can add REMOVE = LINK to such resistor packs without error.

Solution

Remove the REMOVE property from the component. To know more about the REMOVE property, see *Simulation Properties* section of *PCB Systems Properties Reference*.

Incorrect property specified for instance: <instance_name>, pathval <path_value>. The specified property REMOVE = AUTO is being ignored. The REMOVE property can only be used with parts having exactly one input and one output pin.

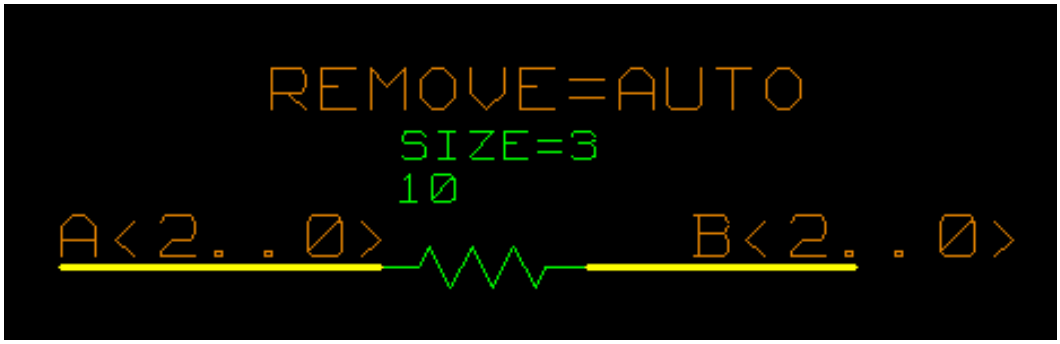
Description

This error is generated when REMOVE = AUTO is attached to a component that does not satisfy the criterion of one input and one output pin. The REMOVE property can be attached only to the two pin components.

Resistor packs with one input and multiple outputs are the only exception to this rule. You can add REMOVE = AUTO to such resistor packs without error.

Note: In most of the resistor packs, REMOVE = AUTO property is assigned at the Origin and cannot be removed.

The REMOVE property also works fine if the size property is used on a two pin part. A part of the schematic that has REMOVE= AUTO attached to a resistor with SIZE = 3 is shown below:



Solution

Delete the REMOVE = AUTO property attached to the part with more than one input or out pin.

Incorrect property value specified for instance <instance_name>, pathval <path_value>. The specified property is being ignored. Supported property values pairs are REMOVE = LINK, REMOVE = AUTO, REMOVE = FALSE, and REMOVE = EXCLUDE.

Description

This warning is generated when the REMOVE property is assigned a value other than LINK, AUTO, EXCLUDE, or FALSE. In this case, Concept HDL netlister ignores the REMOVE property while creating the netlist.

Solution

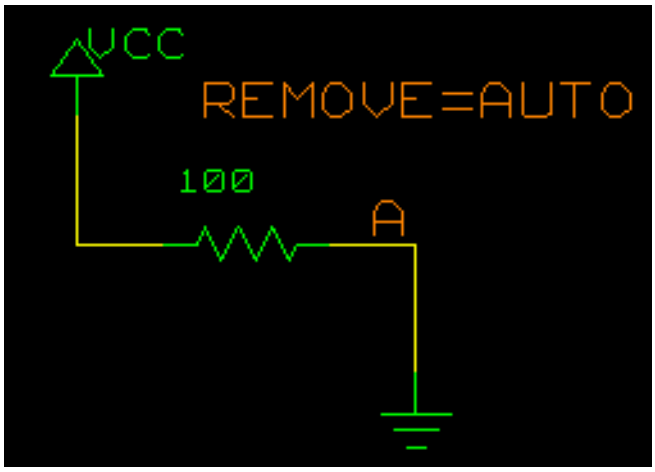
Change the value assigned to the REMOVE property to either LINK, AUTO, or EXCLUDE. To know more about the REMOVE property, see Simulation Properties section of *PCB Systems Properties Reference*.

Power net connections specified for instance: <instance_name>, pathval: <path_value>. One side of the instance is connected to supply 1 and the other side to supply0. The REMOVE= AUTO property can only

be used with parts having one input and one output each of that are connected to different nets. Remove one of the power net connections for this instance from the schematic.

Description

This error is generated when REMOVE = AUTO is attached to a part that is connected to a power source on one side and ground on the other side. Adding REMOVE = AUTO on the resistors shorts the connection between the power supply and ground because of which the error is generated.



Solution

To remove the above error you can do any one of the following:

1. Remove the ground element and ensure that the attached wire has a signal name.
2. Remove the Power supply and ensure that the attached wire has a signal name.

Unable to use PORT_ORDER property for instance: <instance_name>, pathval <path_value>. The directive MAP_BY_POSITION is being ignored.

Description

This warning is generated when you select the *Position Mapping* check box in the *Netlist* tab, and the PORT_ORDER information is not available in the map file corresponding to a part used in the schematic. In this case, the netlist is created using names instead of positions.

Solution

To prevent occurrence of this warning, ensure that the part is bibded to a view that has the PORT_ORDER information in the map fle. To remove the warning, you can:

- modify the map file to add the PORT_ORDER information.
- use SIM_BND_VIEW or SIM_MAP_VIEW property to bind the component to a view containing map file with the PORT_ORDER information.
- clear the *Position Mapping* check box.

Note: To know more about the PORT_ORDER property see *Simulation Properties* in *Concept HDL Digital Simulation User Guide*.

Incorrect property specified for instance: <instance_name>, pathval <path_value>. The property PORT_ORDER is being ignored.

Description

This error is generated when the PORT_ORDER is available in the map file but none of the pins, in the PIN_MAP section of the map file, match to the ports listed in the PORT_ORDER. This error is generated only when the pinlist is NULL. The error will not be generated even if there is a single pin that is mapped to a port listed in the PORT_ORDER list. In the netlist, only those pins that are mapped to some port listed in the PORT_ORDER list appear, rest of the pins are ignored by the Concept HDL netlister.

A part of the map file for a component is shown below:

```
FILE_TYPE=VERILOG_MAP;
...
...
PROPERTY
            PORT_ORDER = '(I1,I2,O1)';
            COMPONENT='SN74LS00';
            RANGE;
        END_PROPERTY;
        PIN_MAP
             'B'<0>='(u2)';
             'A'<0>='(u1)';
             '-Y'<0>='(r1)';
         END_PIN;
    END_MODEL;
END_PRIMITIVE;
```

```
END.
```

In the `verilog.map` file shown above, ports listed in the PORT_ORDER section are I1, I2, and O1. Ports to which the pins are mapping are u2, u1, and r1. As there is no common port, this will generate error.

Solution

Modify the map file to either change the PORT_ORDER information or the PIN_MAP information.

Incorrect property specified for instance: <instance_name>, pathval: <path_value>.The property HDL_REPLICATE is being ignored.

The property HDL_REPLICATE can be added only on instances that have two pins. The input pin must be scalar with pin name INPUT. The output pin must be sizeable like PINNAME<SIZE-1..0>. The SIZE property on the instance will determine the number of times the signal connected to pin INPUT is to be replicated.

Description

This error is generated when you add HDL_REPLICATE = TRUE on a part that either has:

- more that one input or output pins.
- input pin is not scaler.
- output pin is not sizeable.

Solution

Remove the HDL_REPLICATE = TRUE property from the part that does not satisfy the above mentioned criterion. To know more about the HDL_REPLICATE property, see *PCB Systems Properties Reference*.

ERROR 526: In Specifying Split Inst Property on Instance

LOCATION not specified for instance. The specified SPLIT_INST property is being ignored. Specify SPLIT_INST and LOCATION property on the instance

Description

The use model for the SPILT_INST property is that it has to be used with the LOCATION property that specifies the hard location. The value of the location property is then used as the split inst group name.

Solution: Along with SPLIT_INST = TRUE, add the LOCATION property with the same value to all the components that need to form a split inst group. To know more about working with the SPLIT_INST property, see *Working with SPLIT_INST and SPLIT_INST_NAME properties* in *Concept HDL Simulation User Guide*.

Dual SPLIT property specified for instance. The specified property: SPLIT_INST = TRUE is being ignored. Specify either SPLIT_INST_NAME = <value> or SPLIT_INST = TRUE and LOCATION = <value>.

Description

This error is generated when an instance has both SPLIT_INST_NAME, and SPLIT_INST and LOCATION, properties attached to it. You cannot attach both these properties on a single instance. To know more about SPLIT_INST and SPLIT_INST_NAME properties, see *Working with SPLIT_INST and SPLIT_INST_NAME properties* in *Concept HDL Simulation User Guide*.

Solution

Remove either the SPLIT_INST_NAME property or the SPLIT_INST and LOCATION property from the instance.

E

Concept HDL Files

This section describes some of the files that are created or used by Concept HDL.

System Initialization File

The `.cpm` (Cadence Project Manager) file is used to initialize front-end tools that have several setup options as well as tools that require knowledge about the current project.

See the Project Manager online help for more information.

Cadence Library File

The `cds.lib` file contains the list of libraries to be used in conjunction with the project specified in the project file.

See the Project Manager online help for more information.

ASCII Design Data Files

An ASCII design data file is one of the design database files that Concept HDL creates when you write a drawing. This file is read in if the design has no binary design data file (also referred to as a cell). ASCII design data files represent all drawings except symbol (SYM) drawings.

Design data files consist of commands to add each object in a drawing. Concept HDL re-creates a drawing by reading the commands in the ASCII design data file. You can edit the file to modify your drawing.

The Concept HDL name for the ASCII file is SCH with version and page number extensions (for example, `<`*`library`*`>.`*`cell`*`.SCH.1.1.csa`).

ASCII design data files contain:

- File identification and end statements

  Each ASCII logic file starts with this line to identify the file type:

  ```
  FILE_TYPE = MACRO_DRAWING;
  ```

  The file ends with:

  ```
  QUIT
  ```

- Object definitions

Each type of object in a Concept HDL drawing has a specific definition format.

ASCII Object Descriptions For Objects

Components

```
forceadd name
[R angle]
pt ;
[paint color pt]
```

Note: `forceadd` is used so that a placeholder is created if the component is not found. forceadd works only in ASCII drawing files and not in Concept HDL scripts.

name	The component name includes the version number.
R angle	Rotation of the added component. The *`angle`* definition is optional.
pt	Location of the component on the drawing.
paint *`color pt`*	(Optional) Included if the component color differs from the default.

Wires

```
wire linetype pattern pt1 pt2 ;
```

linetype	This numeric argument includes the line color and thickness definition. If the number is converted to binary, the least significant bit is the thin/heavy bit (0 = thin, 1 = heavy). The remaining seven bits specify the color.

pattern	Fill pattern of the line. If pattern is **-1**, the line is filled. Concept HDL has these defined wire patterns: 1-142175 227353135 368264383
pt1 pt2	The begin and end points of the wire.

Dots

```
dot type pt ;
[paint color pt]
```

type	If type is 0, the dot is open; if type is 1, the dot is filled. If the type is neither 0 or 1, Concept HDL assumes the dot is open.
pt	Location of the dot on the drawing.
paint *color pt*	(Optional) Included if the dot color differs from the default.

Circles and Arcs

```
circle pt1 pt2 ; (for circles)
[paint color pt]
or
circle pt1 pt2 pt3 ; (for arcs)
[paint color pt]
```

pt	Location of the circle or arc on the drawing.
paint *color pt*	(Optional) Included if the color of the circle or arc differs from the default.

Notes

```
forcenote contents pt angle;
[display size pt ;]
[paint color pt]
```

Note: The `forcenote` command is similar to the note command in the Concept HDL editor except that the `forcenote` command terminates after reading one note. `forcenote` works only in ASCII drawing files and not in Concept HDL scripts.

contents	Note text.
pt	Location of the note on the drawing.
angle	Rotation of the added note.
display *size pt*	This line is included if the note is not the default size. This command makes the text the correct size.
paint *color pt*	(Optional) Included if the note color differs from the default.

Properties

```
forceprop default_status last name value
[R angle]
[J justification_type]
pt ;
[display size pt ;]
[paint color pt]
```

default_status	Handles changes to properties on library components and can be 0 Unknown status (undefined variable whose status is determined when the component definition is searched) 1 Default (property comes from component definition) 2 Non-default (user-added property)
last	Property is to be attached to the last object or wire entered. If the property is a pin property, the `last` argument is replaced by the `lastpin` argument followed by a *`pt`* that describes the location of the pin in absolute coordinates.
R *`angle`*	Rotation of the added property. The *`angle`* definition is optional.
J *`justification_type`*	Justification of the added text can be 0 Left-justified text 1 Centered text 2 Right-justified text If not specified, the property is created with the current default justification. If an illegal value is given, the property is left justified.

pt	Location of the property on the drawing.
display size *pt*	Sets the visibility of the property name and value.
paint color *pt*	(Optional) Included if the property color differs from the default.

Bubbled Pins

The description of bubbled pins in the ASCII file is

```
forcebubble pt ...
```

All pins that are not in their default bubbled state are listed.

Note: forcebubble works only in ASCII drawing files and not in Concept HDL scripts.

Within ASCII design data files, Concept HDL internal coordinates are 0.00175 inches per unit. Points are represented by their coordinates, enclosed in parentheses and separated by a space. For example, the point x=100, y=200 becomes (100 200).

Angles are represented by a number from zero through seven.

Angle Representation

0	0 degrees
1	90 degrees
2	Mirror of 0 degrees
3	Mirror of 270 degrees
4	180 degrees
5	270 degrees
6	Mirror of 180 degrees
7	Mirror of 90 degrees

Binary Design Data Files

These files contain the same information as the corresponding ASCII file but in a proprietary binary format that is quicker for Concept HDL to read and save. The Concept HDL name for the ASCII file is SCH with version and page number extensions (for example, <*library*>.*cell*.SCH.1. 1.csb).

Symbol Files

These files contain symbol descriptions in ASCII format. Symbol files are saved in an abbreviated format and do not tolerate errors. The system name of the symbol files is SYM with version and page number extensions (for example, <*library*>.*cell*.SYM.1.1.css).

Symbols are composed of:

- Lines
- Arcs
- Text
- Connections
- Component properties
- Pin properties
- Bubble groups

Concept HDL internal coordinates are 0.00175 inches per unit.

ASCII Symbol Descriptions For Objects

Lines

Lines require one line each in the symbol file. The format for a thin line is

```
L x1 y1 x2 y2 [pattern] color
```

The format for a thick line is

```
M x1 y1 x2  y2 [pattern] color
```

x1 y1 x2 y2	The line's endpoint coordinates; the line runs from *(x1 y1)* to *(x2 y2)*.
pattern	(Optional) Identifies the line style (solid, broken, and so on) as a bit pattern. For example, if pattern is -1, the line is solid and if pattern is 682, the line is dotted.
color	Internal Concept HDL color number. The line type describes both the color and thickness of the line. When the integer is converted to a binary value, bit 0 defines the thickness (0 = thin), and the seven most significant bits define the color.

Arcs

`A x y radius start_angle stop_angle color`

x y radius	Center and radius points of the arc.
start_angle/stop_angle	Floating point numbers that measure the angles in degrees counterclockwise from the x axis.
color	Internal Concept HDL color number. The line type describes both the color and thickness of the line. When the integer is converted to a binary value, bit 0 defines the thickness (0 = *thin*), and the seven most significant bits define the color.

Text

`T x y angle slant size over inv just font Nch color string`

Note: The text definition arguments `slant`, `over`, `inv`, and `font` are not currently implemented.

x y	Origin point for the text string.
angle	Angle of the text on the drawing: 0.00 180.00 90.00 270.00
size	Height of the characters.
just	Justification of the added text can be 0 Left-justified text 1 Centered text 2 Right-justified text
Nch	Number of characters and spaces in the string.
color	Internal Concept HDL color number. The line type describes both the color and thickness of the line. When the integer is converted to a binary value, bit 0 defines the thickness (0 = *thin*), and the seven most significant bits define the color.

Connections

`C x y “name dispx dispy bubbleable [default_state x2 y2 x3 y3] f size angle just`

x y	Location of the connection.
“name	Connection name must be enclosed in quotation marks.
dispx dispy	Location of the name.
bubbleable	Whether or not the pin can be defined as low asserted: 0 False 1 True
default_state	Whether or not the pin is low asserted: 0 False 1 True If the default_state is 1 when a component is initially added, the pin is bubbled.
f	Whether the connection is a filled or open dot: 0 Open 1 Filled
size	Size of the name string (default is 41).
angle	Angle of the pin name attached to the connection: 0.00 180.00 90.00 270.00
just	Justification of the string can be: L Left-justified text C Centered text R Right-justified text

Component Properties

`P “name “value x y angle slant size over inv just font NV VV IP color`

“name	Default property name must be enclosed in quotation marks.
“value	Default property value must be enclosed in quotation marks.
x y	Reference point (location) of the property.

angle	Angle of the property.
size	Height of the characters
just	Justification of the property can be: 0 Left-justified text 1 Centered text 2 Right-justified text
NV	Visibility of the property name: 0 Invisible 1 Visible
VV	Property name is visible by default.
IP	Visibility of the property value (default is visible): 0 Invisible 1 Visible
color	Internal Concept HDL color number.

Pin Properties

Pin properties require one line each. They are identical to component properties, except they start with an X instead of a P, and occur directly after the connection with which they are associated.

Bubble Groups

Bubble groups require several lines each in the symbol file. They start with a line that begins with B and end with a line containing only the word END. Each bubble group is on a line by itself in this format:

```
[“name1, “name2, “name3, . . .]
```

All the names are strings with quotation marks. If the bubble group is asymmetrical, the first comma is replaced by a colon.

Connectivity Design Data Files

These files, which are in ASCII format, describe all the components on a drawing, including

- Component names

- Names of signals tied to component pins (including bubble state)
- Component properties

Connectivity design data files are the files on which HDL Direct bases its HDL generation. Connectivity files contain complete path names to the associated library files. The system name of the symbol files is SCH with version and page number extensions (for example, <*library*>.*cell*.SCH_1. 1.csc).

Structure of Connectivity design data files

Header

Example Header in a Connectivity File

```
FILE_TYPE = CONNECTIVITY;
{CONCEPT version and date}
[expression property]
[nets]
[invokes]
END.
```

Optional sections are the

- Expression property
- Net
- Invoke

Note: The continuation character for lines in a connectivity file is a tilde (~). This character can occur anywhere in the line, even in the middle of words, but must be followed by <LF>.

Comments

Comments begin and end with an braces { }. They can appear anywhere in a connectivity file except in the middle of identifiers or quoted strings and can cross lines.

Expression property on a drawing component

Example Expression Property

```
expr property ::= EXPR = expression string;
EXPR=SIZE=10;
```

NET definitions

Each time Concept HDL saves a connectivity file, it numbers all the nets. The NC net is always net zero. Unnamed signals are also numbered. The net numbers are not the same each time the connectivity file is written.

Example Net Definitions in a Connectivity File

```
nets ::= constant “net_name_string [property_list];
```

constant	Net number.
net_name_string	Either the signal name for the net or the unnamed signal string created by Concept HDL; must be enclosed in quotes.
property_list	(Optional) Property name with format: property_list ::= {identifier “string}
identifier	Property name must begin with a letter and can contain only: Letters Digits Underscore (_) Each *property_list* entry must end with a newline character: FILE_TYPE and END.
“string”	The quoted string: 2UN$1$2P$A; 3ANWCLOAD37 CONNECTED_TOPAGE 4;

INVOKE commands to invoke each component in the drawing

Example Invoke Statement in a Connectivity File

```
invokes ::=
% “invoke_name_string
“version_str,xy_str,“rotation,directory_str,path_str;
[parameter_property_list];
[property_list];
```

```
{“pin_name_string [property_list] constant; }
```

invoke_name_string	Component name must be enclosed in quotes.
version_str	Symbol version number must be enclosed in quotes. This property is always output. If this property doesn’t exist, the null string (“ “) is used.
xy_str	Coordinates of the symbol on the page. This property is always output. If this property doesn’t exist, the null string (“ “) is used.

Occurrence Property File (OPF)

The occurrence property file is a database that stores properties for all objects (instances, pins, and nets). It is accessed in read-write mode from both Concept HDL and Packager- XL.

How is an occurrence property file created?

The OPF is created in two ways:

- When you invoke the occurrence edit mode in Concept HDL

  Concept HDL creates an OPF, if it does not already exist (that is created by Packager-XL) and writes all the properties added in the occurrence edit mode into it.

- When you package your design

  Packager-XL creates an OPF if it does not already exist (that is created by Concept HDL). Packager-XL always reads and writes properties into the OPF both in the forward and feedback mode.

If a design has been packaged at least once, the OPF is loaded by Concept HDL in the read-write mode wherein all (relevant) packaging information is accessible. You can modify by using the Attribute menu in the occurrence mode.

Where is the occurrence property file located?

The OPF is located in the opf view of the root cell.

Any property that is attached to an object in the occurrence mode is stored in the OPF at the root level. This is regardless of where they are located in the design hierarchy. The root cell is the design name in the `.cpm` file and can be defined through the Project Manager Setup.

Hierarchy of occurrence property files

Since the OPF always resides at the root level of a design that is being edited or packaged, you can have a hierarchy of OPFs by packaging the hierarchical blocks individually.

OPFs when arranged in a hierarchy where one OPF exists at the top level and the other OPFs exists at a lower level are referred to as Overlayed OPFs and the lower level OPFs are referred to as Subdesign OPFs. The properties in the subdesign OPFs are visible at the top level. In case of a conflict, that is, if a property has different values at the top and lower levels, then the value at the topmost level wins. This is unlike inheritance where the lowermost level always wins.

Property Precedence in OPF

There are two rules for setting the precedence of properties in an OPF:

1. If the same occurrence of an object has a schematic value and an OPF property value, the OPF property value is preferred over the schematic value.
2. If the same occurrence has multiple OPF property values stored at different levels of hierarchy (in different OPF files), then the OPF property value at higher level gets preference over the OPF property value at the lower level.

Inheritance of Properties in OPF

Inheritance is used to propagate properties down in the hierarchy. The OPF handles inheritance in the same way as schematic.

A property, whether schematic or OPF, is propagated to all lower level cells present in the hierarchy. If the same property is present in any lower level cell, the existing property at the lower level cell still gets the preference over the propagated property.

F

The template.tsg File

You can choose *Tools* > *Generate View* in Concept HDL or run the genviewHDL command from the UNIX terminal or the Windows command prompt to generate customized symbols, that is, symbols with certain graphical attributes or properties attached to the pins or the symbol. For more information on using genviewHDL, see the *Concept HDL Online Help*.

genviewHDL obtains the names of pins and properties associated with the symbol and pins of the symbol from the source view or source file. genviewHDL obtains information related to graphical attributes of the symbol and additional pin and symbol properties by reading a template file called template.tsg. Some of the attributes that you can specify using the template.tsg file include the following:

- Pin properties (for example, VHDL_MODE, VHDL_SCALAR_TYPE, and so on)
- Symbol properties (for example, VHDL_GENERIC, LIBRARY, and so on)
- Font size of text used in the symbol drawing
- Color of the symbol box

The default template.tsg file is located at <your_install_dir>/tools/fet/concept/genview/. You can override the specifications in this file by putting your own template.tsg file in the directory from which you start Concept HDL. If genviewHDL cannot find the template.tsg file in either location, it uses hardcoded values for the graphical attributes of the symbol.

Format of template.tsg File

The sections in the template.tsg file and the keywords used in the sections are described below:

defcell

The defcell keyword must be the first keyword in the file. The remaining keywords in the file define the entire symbol. The format of the file is as follows:

```
(defCell <cellname>
    [<defSymbol section>]
)
```

where <cellname> defines the name of the symbol being generated. For example:

```
(defcell "badder"
    ...
    ...
)
```

Note: The name of the generated symbol is obtained from the corresponding name in the source view or source file. For example, if the symbol is generated from a VHDL entity, the name of the symbol is the same as the name of the entity. genviewHDL ignores this keyword.

defSymbol

The defSymbol section describes the properties and attributes of the symbol. This section is defined as follows. Click the links below for more information on the sub-sections of the defSymbol section.

```
(defSymbol
    [<symbolProps>]
    [<symbolParam>]
    [<symbolLabels>]
    [<pinLocSpec>]
    [<pinPosition>]
)
```

symbolProps

The symbolProps sub-section defines the properties that will be attached to the generated symbol and their attributes. Each symbol property can be applied to the symbol, or some or all of the pins on the symbol. This sub-section is defined as follows. Click the links below for more information on the keywords used in this sub-section.

```
(symbolProps
    (defProp
        [(<property name> <property value>)]
        [(apply input|output|io|all|cellview|left|right|top|botton)]
        [(format "off"|"value"|"name=value")]
        [(location (<expr>:<expr>))]
        [(justification left|right|center)]
        [(orientation 0|90|180|270)]
```

```
        [(fontHeight <float>)]
        [(color <colorname>)]
    )
    (defProp
    ...
    ...
    )
)
```

defProp

The `defProp` sub-section defines a property that will be attached to the generated symbol and the set of text attributes that are associated with the property. You must create a `defProp` section for each property that you want to be attached to the generated symbol. This sub-section is declared as follows:

```
    ...
    (defProp
        [(<property name> <property value>)]
        [(<text attribute 1>)]
        [(<text attribute 2>)]
        ...
    )
    (defProp
        [(<property name> <property value>)]
        [(<text attribute 1>)]
        [(<text attribute 2>)]
        ...
    )
```

apply

The `apply` keyword specifies the object(s) to which a property will be attached in the generated symbol. The objects can be:

cellview	attached to the symbol
all	attached to all pins on the symbol
input	attached to input pins
output	attached to output pins
io	attached to io (inout) pins

left	attached to pins on the left
right	attached to pins on the right
top	attached to pins on the top
bottom	attached to pins on the bottom

format

The `format` keyword defines the visibility of the property—whether only the value, name and value, or neither (invisible property) will be displayed in the symbol drawing. The default is “`value`”. The possible format values are:

"off"	Nothing is displayed.
"value"	Only the value is displayed.
"name=value"	Both the name and value are displayed.

location

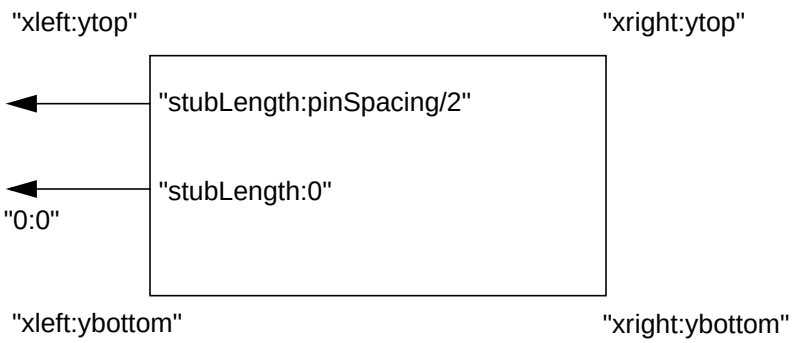
The `location` keyword defines the location where the property will be placed in the symbol. The location must be specified relative to the symbol box or symbol pins. You can define your own locations by forming simple expressions with the following constants:

xleft, xright	Left and right edges of the symbol drawing
ytop, ybottom	Top and bottom limits of the symbol drawing
stubLength	Length of the pin
pinSpacing	Spacing between consecutive pins on any side

For example, you can specify the location using expressions as follows:

"xleft:ytop"	Upper left
"(xleft+xright)/2:(ytop+ybottom)/2"	Center for the symbol properties
"stubLength/2:pinSpacing/4"	Halfway down the pin stub for left pins for pin labels.

The following figure shows the various locations on the symbol, based on some defined constants.



justification

The `justification` keyword defines the justification of the properties placed at the location specified by the location keyword. The possible justification values are `left`, `center` and `right`.

orientation

The `orientation` keyword specifies the orientation of the property. The possible orientation values are:

R0	Horizontal and upright
R90	Vertical facing right
R180	Horizontal upside down
R270	Vertical facing left

The default orientation depends on the location of the object to which the property is attached to. If the property is attached to a pin on the top of the symbol, the default orientation is R90, whereas if the property is attached to the symbol, the default orientation is R0.

fontHeight

The `fontHeight` keyword defines the height of the font used in the properties. The font value is a floating point number. The default font size used by `genviewHDL` is `0.082` inch.

color

The `color` keyword specifies the color that will be used for the property. The possible color values are:

- red
- green
- blue
- yellow
- orange
- salmon
- violet
- brown
- skyblue
- white
- peach
- pink
- purple
- aqua
- gray

symbolParam

The `symbolParam` sub-section specifies the graphical attributes of the symbol. This sub-section is defined as follows. Click the links below for more information on the keywords used in this sub-section.

```
(symbolParam
    (origin topLeft|topRight bottomLeft|bottomRight|center)
    (wireSpacing <float>)
    (wireLength <float>)
    (labelHeight <float>)
    (vSideLength <float>)
    (hSideLength <float>)
    (units inches|mm|cm)
```

```
    (resolution <float>)
    (<color attributes>)
)
```

origin

The origin keyword specifies the location where the ORIGIN symbol will be placed on the generated symbol. The location value can be:

- topLeft
- topRight
- bottomLeft
- bottomRight
- center

wireSpacing

The `wireSpacing` keyword specifies the spacing between the pins of the symbol in user units. If this is not specified, the wire spacing is calculated from the width and height of the symbol and the number of pins. The default wire spacing is equal to the inverse of resolution.

wireLength

The `wireLength` keyword specifies the length of the pin stub. The default wire length equals `1/`resolution.

labelHeight

The `labelHeight` keyword specifies the height of the font used for the labels or notes on the symbol. The font value is a floating point number. The default font size used by `genviewHDL` is `0.082` inch.

vSideLength

The `vSideLength` keyword specifies the length of the vertical sides of the symbol in user units. If this is not specified, the values are computed from the number of pins on the symbol and the pin spacing.

hSideLength

The `hSideLength` keyword specifies the length of the horizontal sides of the symbol in user units. If this is not specified, the values are computed from the number of pins on the symbol and the pin spacing.

units

The `units` keyword specifies the units for all the dimensions specified in the `template.tsg` file. The default value is `inches`.

resolution

The `resolution` keyword determines the round off values for the locations of pins, labels, etc. This should be normally set to the grid spacing of the symbol file. The default resolution is `0.05`.

color attributes

The following color attributes specify the color to be used while displaying the corresponding object.

Attribute	Description
`stubColor`	Color of the pin stub
`boxColor`	Color of the symbol box
`noteColor`	Color of notes on the symbol
`propColor`	Color of the properties
`pinPropColor`	Color of pin properties
`pinNoteColor`	Color of pin notes

The possible color values are:

- red
- green
- blue
- yellow
- orange

- salmon
- violet
- brown
- skyblue
- white
- peach
- pink
- purple
- aqua
- gray

symbolLabels

The `symbolLabels` sub-section defines the labels or notes of the symbol and the set of text attributes that are associated with the label. This sub-section is defined as follows. Click the links below for more information on the keywords used in this sub-section.

```
(symbolLabels
    (defLabel
        [(name “{<labelname>}”)]
        [(apply input|output|io|all|cellview|left|right|top|botton)]
        [(location (<expr>:<expr>))]
        [(justification left|right|center)]
        [(orientation 0|90|180|270)]
        [(fontHeight <float>)]
        [(color <colorname>)]
    )
    (defLabel
    ...
    ...
    )
)
```

apply

The `apply` keyword specifies the object(s) to which a label or note will be attached in the generated symbol. The objects can be:

cellview	attached to the symbol
all	attached to all pins on the symbol
input	attached to input pins
output	attached to output pins
io	attached to io (inout) pins
left	attached to pins on the left
right	attached to pins on the right
top	attached to pins on the top
bottom	attached to pins on the bottom

location

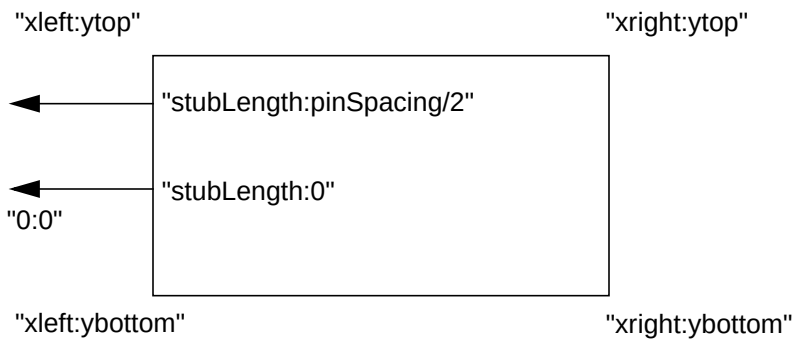
The `location` keyword defines the location where the labels or notes of the symbol will be placed in the symbol. The location must be specified relative to the symbol box or symbol pins. You can define your own locations by forming simple expressions with the following constants:

xleft, xright	Left and right edges of the symbol drawing
ytop, ybottom	Top and bottom limits of the symbol drawing
stubLength	Length of the pin
pinSpacing	Spacing between consecutive pins on any side

For example, you can specify the location using expressions as follows:

"xleft:ytop"	Upper left
"(xleft+xright)/2:(ytop+ybottom)/2"	Center for the symbol properties
"stubLength/2:pinSpacing/4"	Halfway down the pin stub for left pins for pin labels.

The following figure shows the various locations on the symbol, based on some defined constants.



justification

The `justification` keyword defines the justification of the properties placed at the location specified by the location keyword. The possible justification values are `left`, `right`, `center`, `centerRight` and `centerLeft`.

orientation

The `orientation` keyword specifies the orientation of the labels or notes of the symbol. The possible orientation values are:

R0	Horizontal and upright
R90	Vertical facing right
R180	Horizontal upside down
R270	Vertical facing left

The default orientation depends on the location of the object to which the label or note is attached to. If the label or note is attached to a pin on the top of the symbol, the default orientation is R90, whereas if the label or note is attached to the symbol, the default orientation is R0.

fontHeight

The `fontHeight` keyword defines the height of the font used in the labels or notes of the symbol. The font value is a floating point number. The default font size used by `genviewHDL` is `0.082` inch.

color

The `color` keyword specifies the color that will be used for the label or note. The possible color values are:

- red
- green
- blue
- yellow
- orange
- salmon
- violet
- brown
- skyblue
- white
- peach
- pink
- purple
- aqua
- gray

pinLocSpec

This `pinLocSpec` sub-section specifies the side of the symbol where a pin will be placed. By default, input pins are placed on the left side, output pins are place on the right side and io pins are placed at the top of the symbol. If this sub-section is specified, it will override the default pin placement. This section is defined as follows:

```
(pinLocSpec
    (rightPins <pinnames>)
    (leftPins <pinnames>)
    (topPins <pinnames>)
    (bottomPins <pinnames>)
)
```

Where <pinnames> specifies the names of pins whose location is being specified. The pin names are specified in the following format:

```
<pinname 1> <pinname 2>...<pinname n>
```

genviewHDL decides the side for a pin based on its mode in the source view or source file (input, output, inout and so on) and the specification in the pinPosition sub-section. To modify the side on which a pin is placed, Cadence recommends that you modify the pinPosition sub-section.

pinPosition

The pinPosition sub-section specifies the side on which a pin should be placed based on the mode of its corresponding port in the source view. By default, input pins are placed on the left of the symbol, output pins are placed on the right side of the symbol, and io (inout) pins are placed on the top of the symbol. This sub-section is defined as follows:

```
(pinPosition
    (input <side>)
    (output <side>)
    (io <side>)
)
```

where <side> can be left, right, top or bottom.

Pin Name and Cell Name Replacement

Because the template.tsg file can be used to attach properties to a group of pins, you might want to use the name of the pin in these properties. For example, you might want to attach the property <pinname>_DELAY to all the io pins on the symbol. The template.tsg file allows you to specify the name of the property as {pinName}_DELAY (applied to io pins). When the property is generated, genviewHDL replaces the string {pinName} with the actual pin name. Consequently, if there are two io pins, io1 and io2,the io1_DELAY property will be attached to the io1 pin, and the io2_DELAY property will be attached to the io2 pin.

Similarly, if you want to attach a property <design name>_TYPE to all your symbols when they are generated, insert a defProp sub-section with the property name as {cellName}_TYPE. When genviewHDL generates a symbol named badder, it will attach the property badder_TYPE to it.

Sample template.tsg File

The following `template.tsg` file illustrates the use of some of the sections described earlier.

```
(defCell ""
    (defSymbol symbol
        (symbolLabels
            (defLabel
                (name "{cellName}")
                (location "(xright+xleft)/2:(ytop+ybottom)/2")
                (orientation R0)
                (justification center)
                (fontHeight 0.082)
                (apply cellview)
            )
            (defLabel
                (name "{pinName}")
                (location "0:1.15*stubLength")
                (orientation R90)
                (justification left)
                (fontHeight 0.082)
                (apply bottom)
            )
            (defLabel
                (name "{pinName}")
                (location "0:-1.15*stubLength")
                (orientation R90)
                (justification right)
                (fontHeight 0.082)
                (apply top)
            )
            (defLabel
                (name "{pinName}")
                (location "0:-1.15*stubLength")
                (orientation R00)
                (justification centerRight)
                (fontHeight 0.082)
                (apply right)
            )
            (defLabel
```

```
                (name "{pinName}")
                (location "0:1.15*stubLength")
                (orientation R0)
                (justification centerLeft)
                (fontHeight 0.082)
                (apply left)
            )
        )
        (symbolParam
            (origin center)
            (wireSpacing 0.2)
            (wireLength 0.1)
            (labelHeight 0.082)
            (stubColor "green")
            (boxColor "green")
            (pinnotecolor "orange")
            (notecolor "yellow")
            (pinpropColor "peach")
            (propColor "skyblue")
        )
        (pinPosition
            (input left)
            (output right)
            (io top)
        )
    )
)
```

The following notes briefly describe the sample `template.tsg` file:

The name of the symbol:

- Appears at the center of the symbol
- Is justified center
- Is horizontal
- Has a font height of `0.082` inches (the default unit used is inches)

The pins on the bottom of the symbol are labeled. The label for each pin:

- Appears at (`0, 1.15*stubLength`), and the location of the connection is (`0,0`). The label appears slightly above the bottom of the symbol border.

- Is vertically oriented
- Is justified left
- Has a font height of `0.082` inches

The labels for pins on the top, left, and right are defined similarly.

The origin of the symbol will be at the center. The spacing between pins will be `0.2`, and the length of each pin will be `0.1` inches. The font height of the labels and properties associated with the symbol will be `0.082` inches. The pin and the symbol will be green. The pin notes will be orange. The symbol notes will be yellow. The color of pin properties will be peach, and the symbol properties will be skyblue in color.

The input pins (the mode is obtained from the source view, and an input pin corresponds to an input port in the source view) will be placed on the left of the symbol, the output pins will be placed to the right, and the io pins will be placed at the top of the symbol.

Modifying the template.tsg File

You can modify the `template.tsg` file at `<your_install_dir>/tools/fet/concept/genview/` to customize the symbol generated. Alternatively, you can copy the `template.tsg` file to your current directory (the directory from where you start Concept HDL) and then make changes to it. `genviewHDL` first looks for the `template.tsg` file in the current directory. If it does not find the `template.tsg` file in the current directory, it looks for the file in the `<your_install_dir>/tools/fet/concept/genview/` directory.

Adding Symbol Properties

If you need to attach a property to the symbol that will be generated, you can add a defProp sub-section to the `template.tsg` file. This is useful, for example, if you want to attach the `BLOCK=TRUE` property to every symbol you are generating so that you can use the Concept HDL block editor commands on the generated symbol. The following example shows the `defProp` sub-section you need to add:

```
(symbolProps
    (defProp
        (BLOCK "TRUE")
        (format "value")
        (location "(xright:ytop")
        (justification center)
        (fontHeight 0.082)
```

```
    )
)
```

This places the BLOCK=TRUE property at the top right side of the symbol. Only the value of the property (TRUE) will be visible on the symbol. Adding symbol properties using the `template.tsg` file is useful when you need to attach properties to several symbols that you will be generating using `genviewHDL`.

Adding Pin Properties

You can use the `template.tsg` file to attach properties to a group of pins on the symbol. For example, you can attach the VHDL_MODE=IN property to all input pins of the symbol by adding the following `defProp` sub-section in the `template.tsg` file.

```
(symbolProps
    (defProp
        (VHDL_MODE IN)
        (location "-700:0")
        (justification left)
        (apply input)
    )
)
```

Similarly, to attach the VHDL_SCALAR_TYPE=STD_LOGIC property to scalar pins and VHDL_VECTOR_TYPE=STD_LOGIC_VECTOR to vector pins to an existing symbol, include the following section in your `template.tsg` file:

```
(symbolProps
    (defProp
        (VHDL_SCALAR_TYPE STD_LOGIC)
        (apply scalar)
    )
    (defProp
        (VHDL_VECTOR_TYPE STD_LOGIC_VECTOR)
        (apply vector)
    )
)
```

and regenerate the symbol (this can be done by editing the existing symbol and executing `genviewHDL`). `genviewHDL` attaches the VHDL_SCALAR_TYPE=STD_LOGIC property to scalar pins and the VHDL_VECTOR_TYPE=STD_LOGIC_VECTOR property to vector pins on the symbol.

Modifying Symbol Attributes

You can modify the symbolParam sub-section in the `template.tsg` file to change the graphical attributes of the generated symbol. The following `symbolParam` sub-section:

- Places the origin of the symbol at the top left
- Generates a symbol measuring 2 x 3 (height by width in inches)
- Colors the symbol and its pins white
- Colors the notes yellow and the properties red

```
(symbolParam
    (origin topLeft)
    (vSideLength 2.0)
    (hSideLength 3.0)
    (stubcolor "white")
    (boxcolor "white")
    (pinnotecolor "yellow")
    (notecolor "yellow")
    (pinpropColor "peach")
    (propColor "skyblue")
)
```

Changing Pin Positions

By default, `genviewHDL` places the input pins to the left of the symbol, the output pins to the right of the symbol, and the io (inout) pins at the top of the symbol. The mode of the pins (input, output, or io) is determined by the mode of the corresponding port in the source view or source file. To place input pins at the left and output and io pins to the right, add the following pinPosition sub-section in the `template.tsg` file:

```
(pinPosition
    (input left)
    (output right)
    (io right)
)
```

G

Using Concept HDL with PSpice A/D

You can simulate designs created in Concept HDLusing PSpice library components with the PSpice A/D simulator and the PSpice Advanced Analysis add-on program.

PSpice A/D is a simulation program that models the behavior of a circuit. PSpice A/D simulates analog-only, mixed analog/digital, and digital-only circuits. Used with Concept HDL, PSpice A/D is much like a software-based breadboard of your circuit. You can use it to test and refine your design before manufacturing the physical circuit board. For more information, see the *PSpice A/D User's Guide (Concept HDL Version)* .

Note: The PSpice simulator is available in three versions—PSpice A/D, PSpice A/D Basics and PSpice. You can use only PSpice A/D with Concept HDL.

PSpice Advanced Analysis is an add-on program for PSpice A/D. Use these four Advanced Analysis tools to improve circuit performance, reliability, and yield:

- Sensitivity tool allows you to examine how much each component affects circuit behavior by itself and in comparison to the other components. It also varies all tolerances to create worst-case (minimum and maximum) measurement values.
- Optimizer tool allows you to analyze analog circuits and systems. It helps you modify and optimize analog designs to meet your performance goals. Optimizer fine-tunes your designs faster than trial and error bench testing can. Use Optimizer to find the best component or system values for your specifications.
- Monte Carlo tool allows you to predict the statistical behavior of a circuit when part values are varied within tolerance. Monte Carlo also calculates yield, which can be used for mass manufacturing predictions.
- Smoke tool allows you to evaluate component stress due to power dissipation, increase in junction temperature, secondary breakdowns, or violations of voltage/current limits.

For more information, see the *PSpice Advanced Analysis User Guide (Concept HDL Version)* .

Notes for Using Concept HDL with PSpice A/D

Note the following when using Concept HDL with the PSpice A/D simulator and the PSpice Advanced Analysis add-on program:

- You must use components from the PSpice libraries in your design if you want to simulate the design using PSpice A/D. See the *PSpice Library List (Concept HDL Version)* for the list of PSpice library components.
- If you are creating a schematic that you want to simulate using PSpice A/D, you must not use components from the `element` library. Instead, use components from the `pspice_elem` library.
- The components that you want to simulate using the PSpice Advanced Analysis add-on program must be from the PSpice Advanced Analysis libraries. See the *PSpice Advanced Analysis Library List* for the list of PSpice Advanced Analysis library components.
- If you are using the PSpice Advanced Analysis add-on program, ensure that `templates` library is selected for your project. For more information on the procedure for selecting libraries for your project, see Selecting Libraries for a Project on page 78.

  If you are not including the `cds.lib` file located at `<your_install_dir>\share\cdssetup\` in your project `cds.lib` file, you must define the templates library in your project `cds.lib` file as below:

  ```
  DEFINE templates <your_install_dir>\share\library\templates
  ```

  For more information on the `cds.lib` file, see the *Concept HDL Libraries Reference*.

  Note: The `templates` library contains models that are used by the PSpice Advanced Analysis add-on program.

H

Glossary

assertion level

Part of a signal name, it describes the active state of the signal when asserted. By convention, a signal is active high for positive logic and active low for negative logic. An * represents active low - for example, RESET* is an active low signal. Two signals with the same name but different assertion levels are not the same signal.

attribute

Information that Concept HDL lets you attach to objects (components, wires/nets, and pins) in a schematic. Attribute information is passed to other design programs for processing. An attribute consists of a name-value pair. Attributes are also called *properties*. See also *constraint*.

attribute file

A file that contains properties, their associated values, and some display information. Because different types of objects (components, wires, and pins) have different properties associated with them, they need to have different attribute files. A good way to add several properties to an object and ensure their names and values are correct is to use an attribute file as a template. See also *attribute*.

automatic routing

A Concept HDL function that automatically routes wires (*Wire > Route*) around objects in a schematic.

backannotate

The process of updating a Concept HDL schematic with information on new parts, connectivity, and properties from the Design Synchronization and Design Association tools. Usually, you backannotate the design after the first error-free run of Design Synchronization and then again after the design has been processed by a physical design system.

block

A hierarchical representation of a logical collection that can be reused in a schematic.

body

The symbolic representation of a component or design block. This is now called *symbol*.

BODY drawing (symbol)

The symbolic representation of a library component that you add to your design. This drawing defines the shape, pins, and general properties of the library component.

bubble pins

Low-asserted pins represented by circles on pins and indicated with a low-asserted signal name (*).

bus tap

Tapping a subset of signals from a bus. See also *tap.*

bus-through pins

Special pins placed on a component to make it easier to wire a group of components together. Bus-through pins have the same name as the corresponding visible pin.

To find out if a component has bus-through pins, you can use *Display > Pins* to display an asterisk at every pin location.

C-tap body

The default bus tap provided in the Concept HDL standard library.

category

Refers to a group of components arranged hierarchically. Categories can be viewed in Concept HDL with the Component Browser.

cds.lib

A file containing library definitions.

cell

Software representation of a component. Consider a cell to be a collection of views that describe an individual building block of a chip or a system.

chips.prt

A file containing physical information about a component.

component

Refers to the logical characteristics of a library part.

Component Browser

A dialog box in Concept HDL that lists active libraries and their contents, both drawings and components.

component instance

The placement of a component one or more times on a schematic.

configuration

A collection of views that control how a design is compiled and simulated.

connectivity design data file

A file that defines how all the components and nets connect together logically. This file is used by HDL Direct to generate the resulting VHDL or Verilog.

constant signal

A signal that has a numeric name. For example, a signal named 123.

See also, non-constant signal

constraint

A restriction on the physical implementation of a design object.

cross probe

The process of identifying corresponding parts, packages, and signals in the Concept HDL schematic and Allegro layout.

design

A schematic drawing created in Concept HDL.

DOC drawing

A drawing containing only graphics. DOC drawings are used for documentation purposes; no electrical or logical checks are done on them.

DRC

Design Rule Checking.

entity

The view of a cell that contains the definition, including port (pin) definitions, for the current drawing (cell). Several checks are made to ensure that entity declarations, symbols, and schematics are in agreement.

expand

To build a complete design including all levels of the hierarchy based on views specified in the current expansion configuration.

filter

Screens file names, markers, and so on in the current directory and lists only those that match the filter. An asterisk (*) or a blank field lists all the drawings or markers.

flat design

A design in which all parts of the drawing come from Concept HDL or user-defined libraries and are one-to-one logical representations of the physical parts. All of the interconnecting wiring within the design is entered pin-to-pin. Best suited for small designs that do not have sophisticated bus requirements or reuse portions of circuitry.

grid

Defines where wires and pins meet in the schematic. Concept HDL supports three grid types:

- Logic grid for schematic
- Symbol grid for symbol drawings
- Document grid for DOC drawings

hard property

Properties that you add to the schematic to specify packaging assignments. Hard properties are included in the connectivity files and thus also in the Verilog/VHDL netlist. They differ from soft properties, which are essentially documentation properties on the schematic and are not included in the netlist.

hierarchical design

A design that is organized into modules to reuse many of the same circuit functions and isolate portions of the design for teamwork assignments. Using a block design lets you refer to a collection of logic without having to include the logic in the drawing. Hierarchical blocks simplify a drawing. This is also called *block design*.

Hierarchy Editor

A tool to create and edit configurations, which can be used in netlisting. You can also view the components of your design hierarchy using this tool.

injected property

A property that appears to the right of a PPT format definition or part row. Packager-XL passes these properties to Allegro in the physical netlist, - for example, company-specific part numbers, costs, or package types.

in occurrence

A drawing is *in occurrence* when

- Concept HDL understands the hierarchical location of the current drawing page being edited (*Tools > Expand Design*).
- Occurrence editing is enabled (*Tools > Occurrence Edit*) and occurrence properties are loaded into the drawing.

The title bar of the window shows the current page with the notation [in occurrence] - for example, *mycpu.SCH.1.1 [in occurrence]*.

interface signal

A signal property (\I) assigned to pins in block diagrams to indicate an interface signal from a higher level drawing. In a flat design, this is a signal in the schematic that corresponds to a pin in the symbol drawing.

key property

A property that appears to the left of a PPT format definition or part row. Packager-XL uses these properties to uniquely identify the physical part to use from the various table entries. For example, a resistor part table may use VALUE or TOLERANCE to select a specific physical part.

library

A collection of components from which you can select a component to place in a drawing.

library properties

Librarian-generated properties on symbols, chips, and in the Physical Part Table (PPT). Only the librarian can modify library properties.

marker

An error, warning, or information item that indicates a rule violation in your schematic. Markers are generated using the *Tools > Check* menu command, the CheckPlus utility, Design Synchronization, and Packager-XL.

net

A set of pins that are electrically connected.

netlist

An ASCII text file that describes the electrical connectivity (wires/nets and components) of a drawing.

non-constant signal

A signal that has an alphabetical or alphanumeric name. For example, ADDRESS, DATA1, 1CLOCK and so on.

See also, constant signal

NOT body

Used to change the logic convention of a signal. If a signal is asserted low, it is considered to be a negative logic signal. If a signal is asserted high, it is considered to be a positive logic signal. The NOT body is used to change the logic convention of a signal without introducing an actual logical inversion. This implies the state of the signal is not changed, it is just considered to be of the opposite logic convention.

occurrence properties

Properties defined on an object based on its place in the design hierarchy. Occurrence properties are not stored with the Concept HDL drawing but in the Occurrence Property File (OPF) instead. Occurrence properties are manipulated by front-end and back-end designers.

Occurrence Property File (OPF)

A file used by Concept HDL and other tools. It contains occurrence properties. Concept HDL reads occurrence properties from the OPF for the current page and displays them on the drawing. Occurrence properties are saved back to the OPF when occurrence editing is disabled. Other tools also use the OPF.

orthogonal

Bent to route around objects in a schematic. This is an alternative to direct (diagonal) placement.

package

(noun) In VHDL, a collection of types, constants, subprograms, and so on, usually intended to implement some particular service or to isolate a group of related items.

(verb) The process of translating a logical netlist into a physical netlist. Design Synchronization takes a logical representation of a schematic and applies the physical attributes necessary to allow physical layout.

page

Refers to a page in a design. If the amount of logic required to define a design does not fit on a single page, the drawing might extend to more than one page.

part

Refers to the physical symbol derived from the logical representation of a component or design block.

physical

Refers to the physical properties associated with a library component.

Physical Part Table (PPT)

Used to map logical parts in the schematic to physical parts for a layout.

pin

Conductors that protrude from packages. Pins allow the component to be connected logically to wires and other components in the logical design.

placeholder property

A temporary property assigned to the symbol drawing of a part. These properties serve as substitutes for part properties that will be assigned later in the schematic design. Placeholder properties let you predefine the location and text size of part properties through the part symbol drawing.

placeholder value

Substitutes a real property value. It is indicated with a **?** value.

PPT Browser

Lets you select parts based on the properties defined in the PPT file, such as company part number or preferred status.

primitive

The symbol name in the *chips.prt* file.

project

The work area for a design, including all the views of the design, links to libraries, and setup information such as Physical Part Table, configuration, and expansion directives. Separate directories exist for each design project.

property

A logical characteristic of a design object. It is information that Concept HDL lets you attach to objects (components, nets, and pins) in a schematic. Property information is passed to other design programs for processing. A property consists of a name-value pair. Properties are also called *attributes*. See also *constraint*.

ratsnest line

In a design drawing, a line that shows a logical connection between two pins, connect lines, or vias. Elements connected by the same ratsnest line are part of the same net. The ratsnest shows the circuit logic and, for ECL circuits, the order in which pins are to be connected.

reference designator

The designator, or identification code, for a component.

reference library

A library containing cells that describe common components potentially used in many designs.

root drawing

The top-level drawing in your design. This is the drawing that Concept HDL opens by default when you start an editing session.

route

To autoroute a wire (*Wire > Route*). This is an alternative to manually drawing a wire (*Wire > Draw*) around objects on a schematic.

rubberbanding

A feature of interactive commands in which the lines that are attached to an element of the design drawing “stretch” as you move the element with the mouse.

rules-driven design

User-defined design characteristics that can be specified by the schematic (as properties on components, pins, or nets) that are recognized by Allegro and determine processing results.

scalar signal

A signal having a single bit.

SCH drawing

A Concept HDL drawing that contains a *schematic*.

schematic

The standard type of drawing created with Concept HDL to represent the logic of components or design blocks that make up a circuit. The symbolic drawing is generated in a physical layout tool. A schematic can contain library components and design blocks that represent other schematics.

schematic properties

Modifiable properties that are defined when editing the schematic.

scope

You can assign one of three different scopes to a signal:

INTERFACE	Used on signals you want to access from a higher level of a hierarchy. Represented by \I.
GLOBAL	Used on signals that you need to access on all levels of hierarchy. Represented by \G.
LOCAL	Indicates that the signal is recognized only at its own level. No special characters are required because the local scope is the default.

script files

Let you perform repetitive tasks in Concept HDL. You can build a script by editing a file and adding the commands in the sequence you want them to execute. You can use scripts to set up forms for routing, placing, and artwork or executing a series of check plots. Scripts can call other scripts.

section

Refers to a physical section on a logical component. Pin numbers are different in different sections of the component. You can section a component either before or after you package the design.

signal

Wire connections between components that support communication of dynamic data between components. Signals having the same name are interpreted as one signal; this is how signals are connected across multiple pages of a drawing.

signal bits

Signals can have a single bit (scalar signals) or multiple bits (vectored signals). The bit portion of the signal name is called the bit subscript and gives the bit information. Bit subscripts are enclosed in angle brackets, for example, <3..0>.

SKILL

A proprietary Cadence high-level interactive programming language based on the popular artificial intelligence language, LISP.

soft property

Properties that can change from one backannotation to the next. Soft properties are documentation properties on the schematic and are not included in the netlist. They differ from hard properties, which are included in the connectivity files and thus also in the Verilog/VHDL netlist.

structured design

Uses bus signals and memory and register depth. A structured design minimizes the number of interconnections and parts on the schematic.

swap

To exchange the locations of two logically identical pins within a function. This minimizes the average ratsnest crossings in a layout.

SYM drawing

A Concept HDL drawing that contains a *symbol*.

symbol

The symbolic representation of a library component that you add to your design. This drawing defines the shape, pins, and general properties of the library component.

symbol properties

Librarian-generated properties defined on a component through its symbol description and not by editing the schematic.

system properties

Nonmodifiable schematic properties that Concept HDL adds.

tap body

Cadence-supplied taps found in the Concept HDL standard library: C-tap, tap.body, bustap.body, msbtap.body, and lsbtap.body.

text

Includes text can be signal names, properties, and notes.

user-defined net

A net with a signal name on it. Conversely, an unnamed net is one for which the user did not specify a net name and Concept HDL specifies the net name.

vectored signal

A signal having multiple bits.

version

Different graphical but functionally equivalent representations of a component, all of which refer to the same logic drawing. If the version is not specified, Concept HDL assumes the version to be1.

VHDL

VHSIC Hardware Description Language.

view

Designs are represented by these views in Concept HDL: schematic (or logic), symbol (or body), VHDL, and Verilog. Using *Tools* > *Generate View* in Concept HDL, you can generate one view of a design from another.

visibility

Refers to the amount of property or pin information displayed on a schematic.

wire

An electrical connection. A single wire can be an entire net, or, where there are many connections, a wire can be a segment of a net. This is also called *signal*.

wire orientation

Bent to route around objects in a schematic versus direct (diagonal).

Index

Symbols

$XR property 495
$XRERR property 482
.cdsplotinit file 436
    example 436
[] in syntax 26
{} in syntax 26
| in syntax 26

A

Abstract Data Types 167
add command 529
arc command 521
archive
    creating 500
    opening 502
Archiver 499
    overview 499
archiving a project 499
Ascend 108
Assertion 136
attachments
    verifying 519
Attribute File
    Creating 219
Attributes 719
Auto
    Command 532
    Route 534
Automatic Page Borders 90

B

Back annotate
    design 351
Backannotate
    command 611
Basenets report 483
Batch Mode in Windows Plotting 397
batch processes
    running 611
blocks
    hierarchical 455
    non-replicated 456
    read-only 457
    replicated 457
body
    drawings
        in hierarchical designs 517
    origin 519
    PIN NAMES 520
Border
    Placing in the design area 121
braces in syntax 26
brackets in syntax 26
BUBBLE property 521
bubbled pins
    defining 521
bus
    notation 511

C

Case Sensitivity 226
Cell
    Deleting 197
chips.prt file
    Creating 186
circle command
    defining bubbled pins 521
Close window 109
Color 96
    Specifying for individual objects 96
Component List
    Browsing 179
Components 122, 188
    Modifying 189, 341
    Modifying (in a group) 189
    Replacing 190
    Sectioning 196
Concept HDL
    about 27
    changes for in-place cross referencing 457
    console command window 49
    exiting 112
    Getting Started 43

introduction 27
menu bars 46
Modes
Expanded 114
Hierarchy 113
Occurrence Edit 114
non-graphical 611
overview 27
panning 55
starting 43, 118
toolbars 47
user interface 45
zooming 55
Concept HDL Commands
About 54
arc 521
auto property 533
auto undot 533
autodot 532
autopath 533
autoroute 534
backannotate 611
circle 521
Conventions 54
dot 519
move 519
nconcept 611
note 519
pinnames 520
set
default_body_grid 519
show
vectors 511
signame 519
wire 519
write 519
Concept HDL paper sizes 401
concept2cm.log file 315
Console Commands
About 54
Console Window 103
Displaying 103
Editing text 98
constraint management 247
Constraint Manager
auditing obsolete objects 296
Constraint Manager Enabled Flow 263
overview 247
Constraint Manager Enabled Flow
files needed for board layout 267
conventions
user-defined arguments 26
user-entered text 26
creating a new archive 500
Creating Concept HDL Parts 179
Creating Cross References for Power Signals 465
Creating Custom Offpage I/O Flag Bodies 465
Creating the Cref Data File for Page Borders 464
Cref Data File
Changing 471
Cref error report 484
cref.dat file 463, 465, 484
cref.opf file 457
CREF_FROM_LIST 488
CREF_ORIG_DESIGN_NAME 488
CREF_ORIG_PAGE 488
CREF_ORIG_VIEW 488
CREF_TO_LIST 488
CRefer
overview 449
Crefparts report 484
Cross probe
Overview 350
Cross Reference
Reports 483
Types 450
cross references
flat 450, 451
hierarchical 450, 452
types 450
Cross-Referencing
Determining the Right Cross Referencing Options 463
Cross-Referencing the Design 61, 470
Cross-View Checking 636
CSF Mechanism 355
CTAP 618
CURRENT_DESIGN_SHEET 488, 491
custom text 462
Custom Variables
About 239
Case-Sensitivity 246
Defining 243
Inbuilt 240
Types 239
custom variables 487

D

DECLARATIONS 616
default
  body
    drawing grid 519
  grid setting 519
defining
  bubbled pins 521
  low-asserted pins 521
Defining a Default Text Editor 90
Defining a Text Macro in a File 234
Defining Output Reports 476
Defining Physical Property Options 191
Defining Text Macros on a Drawing Using the DEFINE Symbol 233
Defining Text Macros Using or 234
Design
  Cross-Referencing 61
Design Techniques
  flat 507
  hierarchical 515
  Overview 507
  structured 510
Determining Coordinates 466
Dialog boxes
  Editing text in 98
dialog boxes
  CRefer 471
  Cross Referencer Options - Content Tab 473
  Cross Referencer Options - Cref Data File Tab 472
  Cross Referencer Options - Format Tab 475
  Cross Referencer Options - Reports Tab 477
directives
  NONTRIVIALNET 465
  TRIVIALNET 466
Display schematic information 99
dot command
  in body drawings 519
DRAWING 625
Drawing
  BODY 517
drawing
  names
    multiple page drawings 509
  pages 508
Drawings 103, 107, 121, 197
  adding PIN NAMES bodies 520
  Ascending 108
  Deleting cells
    views
      and files 197
  Descending 108
  Going to previous and next page 103
  LOGIC 517
  multiple page 509
  Panning 107
  Placing a border 121

E

edbconfig Command 140
electrical constraints
  capturing 272
  constraint differences 297
  deleting 295
  in hierarchical designs 287
  in read-only blocks 290
  in replicated blocks 290
  modifying 293
  on bits of buses 276
  on buses 275
  on pin-pairs 282
  overview 247
  placeholder for 291
Entity Declaration 636
  Generating 183
Error status bar
  Displaying 103
Expanded Mode 114

F

Files
  .cdsplotinit 436
  cds.lib
    Adding Libraries 77
  chips.prt 186
  Created for New Project 70
  Deleting a File 197
  part table file 187
  pptoptionset.dat 192
  xmodules.dat 357
files
  cmbcview.dat 266

cmdbview.dat 266
compview.dat 261
concept2cm.log 315
cref.dat 463, 465, 471, 484
cref.opf 457
funcview.dat 261
netview.dat 261
pinview.dat 261
pstchip.dat 261
pstcmback.dat 315
pstcmbc.dat 264
pstcmdb.dat 264
pstxnet.dat 261
pstxprt.dat 261
Finding nets and cells 110, 344
FLAG 631
flag bodies 465, 480, 482
flat cross references 450, 451
Flat designs 507
Creating 508
flattener
schematic 515
Formatting Options
Configuring 474
Front-to-Back Flow
Constraint Manager Enabled Flow 263
Traditional Flow 260

G

Global Find
Limitations 345
Grid
Displaying 103
grid
default settings for bodies 519
Groups 331, 332
Creating (by expression) 332
Overview 331

H

HDL DECS 616
HDL_PORT 468
HDL_PORT property 450
hier_write 469
Hierarchical
designs 507, 515
hierarchical block 455
hierarchical cross references 450, 452
Hierarchical designs
creating 516
Hierarchical Plotting 424
Hierarchy Mode 113
highlighting 61

I

I interface signal property 518, 520
I/O Types 486
Identifying Inputs and Outputs 479
Inbuilt Custom Variables 240
in-place cross referencing algorithm 460, 461
interface signal property (I) 518, 520
introduction to Concept HDL 27
italics in syntax 26

K

keywords 26

L

Lib-Cell-View 64
Libraries
Standard 613, 632
Library Clauses 185
Library components
Deleting 197
LIBRARY Property 185
Limitations of Global Find 345
literal characters 25
LOGIC drawings 517
Logic Type
Verilog 157, 159
VHDL 160
low-asserted pins 521
LSBTAP 619

M

Making Cross References Permanently Visible 467
Merge 153
Modes in Concept HDL 113

Module Ordering 351
module ordering 462
Modules
  Global exclusion 354
Move
  Grouped Objects 334
move command 519
MSBTAP 620
multiple
  page drawings
    flat designs 508
    names 509
    signals 509

N

names
  drawing
    multiple page drawings 509
  signal 509
nconcept command 611
netist
  for simulation 375
netlist
  disable netlisting 374
  for analog and mixed signal simulation 382
  for digital simulation 376
  for packaging the design 372
  for synthesis 378
  packaging 372
  PSpice 382
Nets
  Navigating in a design 111, 346
Netsbypage report 483
Next page 103
non-graphical Concept 611
NONTRIVIALNET directive 465
NOT 631
notation
  bus 511
note command
  labelling body drawings 519

O

Occurrence Edit 237
  Enabling/disabling 237
  Viewing occurrence properties 237
Occurrence Edit Mode 114
Occurrence properties
  Viewing 237
OFFPAGE flag body 480, 482
OFFPAGE property 468
OPF 487
Or-bars in syntax 26
origin
  body 519
Overview 449

P

Page 197
  Deleting 197
  Renumbering 364
Page Borders
  A Size Page 624
  Automatic 90
  B Size Page 624
  C Size Page 624
  D Size Page 624
  E Size Page 624
  F Size Page 624
Page Numbering 487
page numbering 487
Page Renumbering
  About 364
  Commands
    page delete 366
    page forcereset 367
    page move 366
    page reset 367
    page swap 366
  Using 365
pages
  drawing 508
pan schematic 107
Paper Sizes
  Supported by Concept HDL 401
Part Table File
  About 79
  Adding 80
Part table File
  Creating 187
parts
  sizeable 511
  standard library 515
PIN NAMES 630
PIN NAMES body 520

pinnames command 520
pins
  bubbled
    defining 521
  changing states 194
  low-asserted 521
placeholders
  adding 495
  support 459
Plot command 400
plotter
  specifying a 436
Plotting
  Hierarchical Plotting 424
  Windows 387
  Windows Plotting 387
Port names
  Adding from corresponding symbol 223
Ports 132
  Verilog Logic Type 157, 159
  VHDL Logic Type 160
pptoptionset.dat file 192
Preparing the Design for Cross Referencing 461
Pre-Select Mode 90
Previous page 103
Previous window
  Displaying 103
procedures
  adding offpage symbols 468
  adding placeholders on ports or offpage symbols 495
  changing the cref data file 471
  configuring formatting options 474
  configuring run and write options 472
  creating cross references for power signals 465
  creating custom offpage I/O flag bodies 465
  creating the cref.dat file for page borders 464
  defining output reports 476
  deleting cross references 478
  determining coordinates in Concept HDL 466
  determining the right cross referencing options 463
  editing of invisible placeholders 497
  generating cross references for a design 470
  making cross references permanently visible 467
  managing changes in the standard library 498
  performing to/from property annotation 489
  preparing the design for cross referencing 461
Project Files 65
Project Flows 68
Project-Specific exclusion of modules 354
Properties 223, 237
  Reattaching 223
  Swapping 223
  Viewing occurrence properties 237
properties
  $XR 495
  $XR0 495
  $XRERR 482
  BUBBLE 521
  hard 458
  HDL_PORT 450, 468
  OFFPAGE 468
  SIG_NAME 459
  signal
    interface (I) 520
  SIZE 510
  TIMES 512
  XR 459
Property Options 191, 192
ptf file 79
  Adding to a Project 79

R

Ranges
  Ascending 164
  Descending 164
  Syntax 163
  Unconstrained 165
Redo 92
Renumbering Pages 364
REPLICATE 626
replicated blocks 457
reports
  Basenets 483
  Cref error 484
  Crefparts 484
  Netsbypage 483
  part cross reference 484
  signal cross reference 484

Synonym 483
Resolved Types 165
Rotate a component 195
Run and Write Options
Configuring 472
running batch processes 611

S

Sample Cref Data File 485
Sample Signals Labeled with Cross References 481
saving the design 170
Scale a drawing 107
schematic flattener 515
Schematic Reports 484
Scripts 348
Running 348
Selecting a Property File 82
Selecting a Text Editor 82
set command
default_body_grid 519
Setting Up a Log File 83
show command
vectors 511
SIG_NAME property 459
signal
names
multiple-page drawings 509
properties
interface (I) 518, 520
syntax 511
signal I/O types 486
Signals 616
Assertion Level 136
Global 140
Initial Value 135
Merge 150
Naming 126
Tapping 618, 619, 620
Unnamed 147
Verilog Logic Type 157, 159
VHDL Logic Type 160
Signals that are Not Cross Referenced 482
signame command
in body drawings 519
SIM DIRECTIVES 632
single-page flat designs
creating 508
SIZE property 510
sizeable
parts 511
SLASH 135
Slice 154
Spin a component 196
Standard Library 613, 632
Standard library
Overview 175
standard library 515
starting
Concept HDL 43, 118
Status bar
Displaying 103
strict entity check 380
Strokes
Running commands with 112
Structured designs
About 507, 510
Creating 510
SUPPLY_0 627
SUPPLY_1 628
Support for Occurrence Property Data 487
Suppressing Cross Referencing of Signals 465
Symbols
Creating 182
SYNONYM 197, 622
Synonym report 483
SYNOP DEC 615
syntax
signal name 511

T

Tap 198, 616
Template File
Changing 471
Text 220, 221, 222, 223
Adding 221
Adding port names from corresponding symbol 223
Changing the editor 222
Editing in dialog boxes and the console window 98
Modifying 221
Resizing 222
Swapping 223
Text Editor
Changing 222
Textsize Command 600

TIMES property 512
Toolbars 104
  Displaying 103
Tools Setup 81
TOTAL_DESIGN_SHEETS 488, 491
Traditional Flow 260
  files needed for board layout 263
Transcribe 515
TRIVIALNET directive 466
Type Conversion 166

U

UI options
  Add Crefs as Hard Properties check box 458
  Cref Signals Not Connected to Flag Bodies check box 462
  Cref Signals Not Connected To Flagbodies check box 458
  Generate Flattened Schematic 452
  Generate Flattened Schematic check box 452
  Ignore Inputs Only Signals check box 482
  Omit Hierarchical Arrows 453
  Omit Input/Output Arrows check box 452, 479
  Omit Zone Information check box 452
  Redo Placement of Crefs check box 458, 463
  Redo placement of crefs check box 459
  Show Block Names in Hierarchical Cross References check box 454
  Show Signal Names in Hierarchical Cross References check box 454
  Show Warnings for Unique Signals check box 482
Ui options
  Omit Hierarchical Arrows check box 479
Unbubbled pins
  defining 521
Undo 92
Use
  Clause 185
  Property 185

V

verbose output 380
verifying attachments 519
Verilog Net Type 157, 159
VERILOG_DECS Symbol 613
vertical bars in syntax 26
VHDL Scalar Type 160, 162
VHDL Vector Type 160, 162
VHDL_DECS Symbol 613
VHDL_IN_CONVERT 166
VHDL_INIT 135
VHDL_OUT_CONVERT 166
View 197
  Deleting 197
View Names
  Changing 87

W

Windows
  Closing 109
  Displaying previous 103
  Displaying Toolbars 104
  Moving 109
  Resizing 109
Windows Plotting 387
  Batch Mode 397
  Previewing 388, 392
  SetUp 388
wire command
  marking clock signals 519

X

XR property 459
X-Replication 168

Z

zoom 55, 107